

AD-A105 423

OHIO STATE UNIV RESEARCH FOUNDATION COLUMBUS  
DISCRETE CONTROL MODELLING OF SMALL TEAMS.(U)  
JUN 81 R A MILLER

F/6 9/2

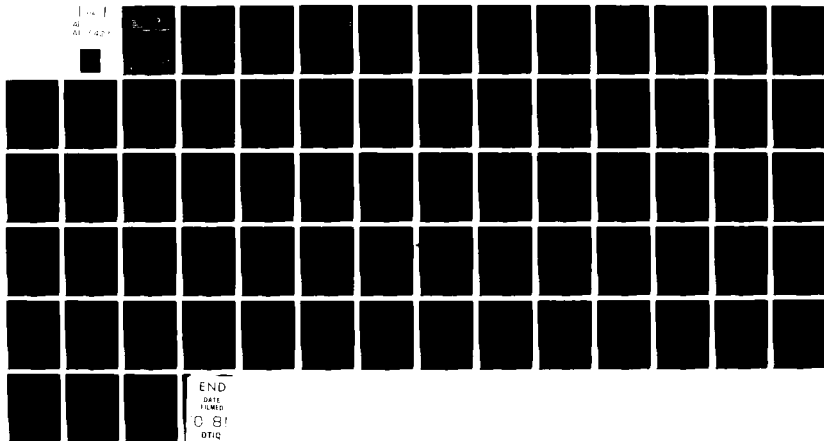
UNCLASSIFIED

AFOSR-TR-81-0655

AFOSR-79-0015

NL

1 of 1  
AD  
AD-A105 423



AFOSR-TR. 81-0655

RF Project 761348/711544  
Final Report

LEVEL

AD A105423

the  
ohio  
state  
university

DTIC

OCT 9 1981

H

(2)

research foundation

1314 kinnear road  
columbus, ohio  
43212

DISCRETE CONTROL MODELLING OF SMALL TEAMS

Richard A. Miller  
Department of Industrial and Systems Engineering

For the Period  
October 1, 1978 - January 31, 1981

U S. AIR FORCE  
Air Force Office of Scientific Research  
Washington, D.C. 20332

~~AFOSR-79-0015~~ AFOSR-79-0015

June, 1981

DISTRIBUTION STATEMENT A  
Approved for public release  
Distribution Unlimited

DTIC FILE COPY

81 10 5 021

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER <b>AFOSR-TR-81-0655</b>	2. GOVT ACCESSION NO. <b>AD-A205423</b>	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) <b>DISCRETE CONTROL MODELLING OF SMALL TEAMS.</b>		5. TYPE OF REPORT & PERIOD COVERED <b>Final</b>
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) <b>Richard A. Miller</b>		8. CONTRACT OR GRANT NUMBER(s) <b>AFOSR-79-0015</b>
9. PERFORMING ORGANIZATION NAME AND ADDRESS <b>The Ohio State University Research Foundation, 1314 Kinnear Road Columbus, Ohio 43212</b>		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS <b>611021 2312/1A4</b>
11. CONTROLLING OFFICE NAME AND ADDRESS <b>U.S. Air Force Air Force Office of Scientific Research/NL11 Washington, D.C. 20332</b>		12. REPORT DATE <b>June, 1981</b>
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) <b>16, 71</b>		13. NUMBER OF PAGES <b>71</b>
		15. SECURITY CLASS. (of this report) <b>Unclassified</b>
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)  <b>Approved for public release; distribution unlimited.</b>		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)  <b>finite state systems, markov modelling, computer integrated display design, human-machine systems modelling</b>		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)  <b>Three basic research directions were pursued. The first was the con- struction of computer aids which enable an analyst to interactively structure and specify a discrete control model. The resulting model can be used to automatically analyze a data base containing event oriented operator perfor- mance data. The second task involved providing theoretical and analytical support for the construction of a multi-level, discrete control based model of planning related activity in a target acquisition task. The third effort (continued)</b>		

DD FORM 1473  
1 JAN 73

Unclassified


SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

Block #20 (Abstract) -- Continued

examined the use of discrete control concepts in the design of computer integrated information display systems. This effort demonstrated, in part, the usefulness of discrete control when structuring a problem in such a way that operator oriented displays can be designed.



Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

DISCRETE CONTROL MODELLING  
OF SMALL TEAMS

12

DTIC

OCT 9 1981

H

by

R. A. Miller  
Dept. of Industrial and Systems Engineering  
The Ohio State University  
Columbus, OH 43210

For the Period:  
October 1, 1978-January 31, 1981

U.S. Air Force  
Air Force Office of Scientific Research  
Washington, D.C. 20332  
 AFOSR-79-0015

June 1981

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)  
NOTICE OF TRANSMITTAL TO DTIC  
This technical report has been reviewed and is  
approved for public release IAW AFR 190-12.  
Distribution is unlimited.  
MATTHEW J. KERPER  
Chief, Technical Information Division

Approved for public release,  
distribution unlimited.

# ABSTRACT

Three basic research directions were pursued. The first was the construction of computer aids which enable an analyst to interactively structure and specify a discrete control model. The resulting model can be used to automatically analyze a data base containing event oriented operator performance data. The second task involved providing theoretical and analytical support for the construction of a multi-level, discrete control based model of planning related activity in a target acquisition task. The third effort examined the use of discrete control concepts in the design of computer integrated information display systems. This effort demonstrated, in part, the usefulness of discrete control when structuring a problem in such a way that operator oriented displays can be designed.

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Avail and/or	
Dist	Special
A	

## TABLE OF CONTENTS

Abstract		i
List of Figures		iii
List of Tables		iv
Chapter		
1	Introduction	1
2	Interactive Computer Aids	3
2.1	Discussion of the Method	4
	Representation By Tables	5
	The Use of Logical Operations	6
	Example	9
2.2	Outline of the Program	13
	Naming Subsystems	14
	Input-Output Organization	14
	Primitive Inputs To Data Analysis	20
	Output of the Program	20
	Computer Programs	22
3	Discrete Control and the Design of Integrated Displays	29
3.2	System Dynamics, Controls, and Constraints	31
3.3	The Experimental Conditions	36
3.4	The Control Station and Information Display Console	37
3.5	Data, Performance Measures and Incentives	40
3.6	Results	42
3.7	Discussion of the Results	52
References		63

## LIST OF FIGURES

Figure		Page
2-1	Function of the General Program	4
2-2	Function of the Structural Representation of Subsystems	5
2-3	Examples of Logical Functions	8
2-4	Termination Node	8
2-5	Example of Graphical Representation of Subsystems	11
2-6	Subsystems in the Network and the Network Representation	15
2-7	Network Representation of Example System	16
2-8	Example of Input File (2): Structural Representation of the Example Subsystem	17
2-9	Graphical Representation of Example Subsystem	18
2-10	Primitive Variables	21
2-11	Subsystem States	23
2-12	Structure of Routines	24
2-13	Flowchart of STATE	25
3-1	Schematic of the System	32
3-2	Mean Score Per Session By Display Type	43
3-3	Mean Number of Engines Processed Per Day By Display Type	45
3-4	Mean Number of Expedited Engines Processed Per Day By Display Type	46
3-5	Mean Number of Errors Per Day By Display Type	47
3-6	Mean Percentage of Time an Engine Was Engaged In Testing Per Day By Display Type	49
3-7	Mean Percentage of Time an Engine Was Held In Station 6 Per Day By Display Type	51



## LIST OF TABLES

Table		Page
2-1	Table Representation of Subsystems	6
2-2	Table Representation of Example Subsystem	9
3-1	Comparison of Display Attributes	38

## 1. INTRODUCTION

Research performed under the sponsorship of this grant falls into three main categories:

- 1) continued development of computer aids for discrete control analysis
- 2) development of a three level, finite state representation of a target acquisition task with continuous control variables
- 3) an exploratory examination of the use of discrete control modelling in the design of computer integrated displays.

These areas of research all directly contributed to the main theme of the grant, namely, attempting to better understand how operators organize their knowledge of complex systems and how they process information while performing their assigned tasks.

Issues explicitly associated with teams were not directly pursued for several reasons. Since the grant was for one year of research (a two year program was originally proposed) no experimental or empirical work could be directly accommodated. When the opportunity to take advantage of data from two experiments not directly funded by the grant became available, it was determined that progress toward the long range goal of understanding how operators organize complexity would be best served by concentrating on the available data rather than attempting more speculative and exploratory research.

The research task on computer aids for discrete control analysis developed interactive techniques which an analyst can use to define the network structure and the constructive specification of each node in a discrete control network. The resulting structure can then be used to automatically analyze a data base containing operator behavior data, or to simulate operator performance.

Prior to this work a major programming effort was required each time a new model structure was considered. This research is discussed in section 2.

The work with the target acquisition data was particularly interesting since the data were not inherently discrete. The experiment was conducted by Dr. Richard Jagacinski with funding provided by Grant No. AFOSR-78-3697.

The effort supported by this grant consisted of the theoretical and analytical support necessary to develop and parameterize a discrete control model. The results of this effort are discussed in Jagacinski, et. al., 1981 and therefore are not reviewed here.

The third research task was quite different from all previous discrete control related efforts. Here, rather than analyzing data, discrete control techniques were used to represent an operator's information processing task in a hypothetical system and then design a display and information system which accounts for the operator's information needs at a reasonable level of abstraction. Funding from this grant was used to support the system analysis and display design. The experiment was conducted with funding provided from other sources. The problem and a summary of the experiment and the results are presented in section 3.

## 2. INTERACTIVE COMPUTER AIDS

Discrete control is a type of manual control in which the operators have a finite number of control or decision alternatives to control the system behavior over time. Miller (1979) used finite state methods to model discrete control task performance and showed that stochastic automata could be used to parameterize the model. The behaviors of complex systems can be explained and important decision points can be identified with the discrete control methods developed.

The systems that are analyzed by finite state modelling can be very complex and the number of parameters involved is generally large. To analyze the system as a whole is quite difficult and sometimes impossible. This is one of the reasons why the system is decomposed into small subsystems at the beginning of the analysis. The analyst defines the subsystems according to the specific purpose of the study. The subsystems are then linked together to form a network. This network representation of the system is a critical step in finite state modelling.

This research is an attempt to develop computerized analysis aids to simplify the construction of the network mentioned above. An interactive program which allows the analyst to define subsystems and the system network, and then automatically compute system state transition behavior over time is the objective.

Every complex system can be represented by simpler subsystems that are connected to each other forming a network structure. Each subsystem is defined by a set of inputs, a set of outputs, and a state representation. The subsystems transfer information along the arcs of the network. The inputs to a subsystem in general are the outputs of other subsystems, and the outputs of that subsystem are inputs to others. It is possible then to represent the structure of a complex system simply as a directed graph

with nodes consisting of the subsystems and the arcs consisting of communication links. A network representation of the system is constructed in this manner at the initial stages of discrete control modelling. Typically computer routines are then written to analyze the behaviors of the subsystems over time. These routines are specific to the network developed; that is, different programs must be written for different representations. If the analyst wants to make changes in the network, he has to rewrite the computer programs to correspond to the new representation. This is a time-consuming process because it is likely that the best model of the system will not be the first one, and as the model is modified the programs used must be changed. The need for a general computer program that can be used to analyze a large class of networks, rather than one specific network, motivated this study.

## 2.1 Discussion of the Method

Figure 2-1 shows the function of the required program. It must compute the states of all the subsystems in the network. The program output, subsystem state information, must be in events form.

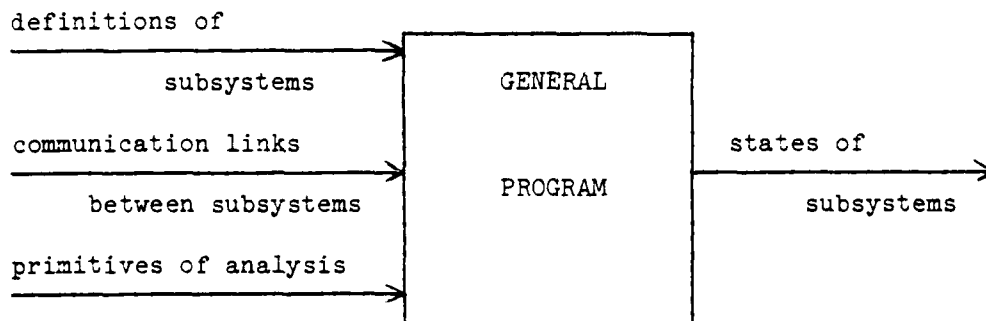


Figure 2-1. Function of the General Program

As shown in Figure 2-1, one class of inputs to the general program is a primitive inputs data base. This is also event based data. Another input to the general program is the definition of the state computation rules for each subsystem. The last input required is the definition of the communication links between the subsystems, i.e. the network structure.

Figure 2-2 shows a key function of the general program described above. This function assigns the state of a subsystem using the inputs to that system. Using this function repeatedly, all the subsystem states can be computed. But the order of this computation is important and is controlled by the structural representation of the subsystem.

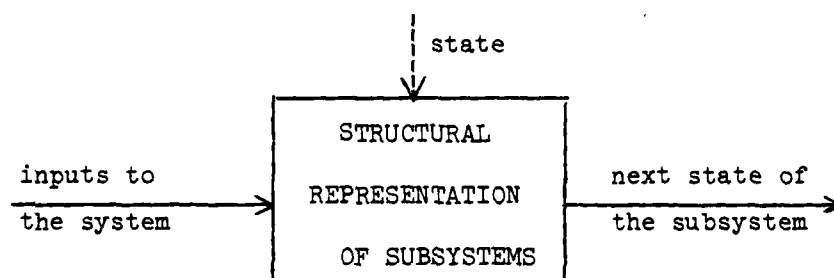


Figure 2-2. Function of the Structural Representation of Subsystems

The state input is shown with a dotted arrow because it is not used in combinational subsystems; only sequential subsystems require that input.

#### Representation by Tables

The representation of subsystems can be structured in various ways. Table 2-1 shows a simple table representation of a system with three inputs, namely  $I_1$ ,  $I_2$ ,  $I_3$ . It is similar to a truth table except that the values of the inputs and the state of the system are not necessarily binary numbers.

I1	I2	I3	state
1	1	1	1
1	2	1	1
1	1	2	2
1	2	2	3
2	1	1	3
2	2	1	2
2	1	2	2
2	2	2	3

Table 2-1. Table Representation of Subsystems

In this table, input values of the subsystem form the columns of the table and the different combinations of inputs are shown in the rows. The last column is the state value which corresponds to the combination of inputs in that row. So the first column in Table 2-1 can be interpreted as "if I1 = 1 and I2 = 1 and I3 = 1, the system will be in state 1". In general tables will be very long since there is a row for every combination of input values. In Table 2-1 there are three inputs each with two possible values, so the number of combinations is  $2^3 = 8$ . A system with three inputs with four values each could require eighty-one rows! Clearly, tables would be a very difficult mechanism.

#### The Use of Logical Operations

The representation of subsystems by tables is not adequate for our purpose since the tables would be very long and not convenient for computer applications. Before putting more structure on the representation of subsystems, certain logical operations will be defined. These logical operations will be used to summarize the information given in tables similar to

Table 2-1. Subsystems will be represented using this set of logical operations as binary trees which are suitable for computer applications.

Three operations are used in this discussion: AND, OR, SET. Additional operations can be added if required. The first two, AND and OR, have two operands each. AND, OR operations with more than two operands can be defined similarly if desired. The AND operation is represented by a " $\wedge$ " sign. In the AND operation if the first operand has a specified value and the second operand has a specified value the result of the operation is "true". If either of the operands does not have the corresponding specified values, the result of the operation is "false". The OR operation is represented by a " $\vee$ " sign. In an OR operation, if the first operand has a specified value or the second operand has a specified value, the result of the operation is "true". If both of the operands do not have the corresponding specified values, the result of the operation is "false". The SET operation checks if a given element is in a specified set. The element is compared with the elements of the set one by one. If the given element is identical to any of the elements of the set, the result of the SET operation is "true", if not the result is "false".

Logical functions are written using the three logical operations defined above. The value of a logical function is either one or zero. One stands for "true" and zero stands for "false". Figure 2-3 shows examples of logical functions that use the logical operations defined. The first function uses an AND operation. The value of this function is one if I1 equals three and I2 equals one. Otherwise the value of this function is zero. The second function uses an OR operation. The value of the function is one if I3 equals two or I1 equals two. If I3 does not equal two and I1 does not equal 2, the value of the function is zero.



The third function in Figure 2-3 uses a SET operation. It checks if I2 is in the set (1,2,4,6). If I2 equals one or two or four or six, i.e. is one of the elements of this set, the value of this function will be one. If not, the value of the function will be zero. The SET operation can be formed by repeated OR operations and is therefore not strictly necessary. It is included to simplify the analyst's modelling task.

$$I3 = 3 \wedge I2 = 1$$

$$I3 = 2 \vee I1 = 2$$

$$I2 \in (1,2,4,6)$$

Figure 2-3. Examples of Logical Functions

In addition to the logical operators, output assignment operations as shown in Figure 2-4 are defined. They assign a "value" to the state of the subsystem.

$$\text{state} = \text{value}$$

Figure 2-4. Termination Node

The logical functions defined above and the tree graphs give the necessary structure to construct subsystem state computation rules. Trees can be constructed with logical functions and termination boxes as their nodes (vertices) to represent subsystems. This is illustrated by the following example.

### Example

Table 2-2 shows an abbreviated table representation of the subsystem which will be used as an example. The structure of this table is a little different than the one shown in Table 2.1. To shorten the length of the table some rows can be combined. This is done again by using logical symbols:

V "or"

$\wedge$  "and"

$\neg$  "not"

As  $\neg(1 \wedge 2)$  means neither one nor two,  $1V2V3V4$  means either one or two or three or four.

The subsystem in Table 2-2 has four inputs: I1, I2, I3, I4. The inputs have three, four, two, and six values respectively. States of the subsystem for different combinations of inputs are given in the last column. There are four states for this system.

INPUT 1 (I1)	INPUT 2 (I2)	INPUT 3 (I3)	INPUT 4 (I4)	STATE OF SUBSYSTEM
3	1	1V2V3	1V2V3V4V5V6	1
3	$\neg 1$	1V2V3	1V2V3V4V5V6	4
3	1V2V3V4	1V2V3	$\neg(1V2)$	1
$\neg 3$	1V2V3V4	1V2V3	2	2
$\neg 3$	1V2V3V4	$\neg 3$	1	3
$\neg 3$	1V2V3V4	3	1	4

Table 2-2. Table Representation of Example Subsystem

The task is to compute the information given in Table 2-2 using logical operations. If the Table 2-2 is examined carefully, it can be seen that when the values of I1 and I2 are three and one, respectively, the output is one no matter what the other inputs are. Similarly, if the inputs I1, I2 are three and one respectively, then the output is four no matter what the other inputs are. These facts define the required computational steps.

Figure 2-5 is a computational representation of Table 2-2. Each decision box has two outgoing arrows labelled with a one or zero. One of the two will be followed to the next node according to the result of the logical operation. Terminating nodes have no outgoing arrows.

Figure 2-5 is a binary tree. Since there can be two results at each node, either one or zero, the graphs of this application are binary trees. If operations requiring more than two possible outcomes are defined, the binary character would not apply, but the tree structure would be preserved.

The procedure of computing the subsystem state starts with the uppermost node which is "I1 = 3 AND I2 = 1". This can be translated as "if I1 has a value three and I2 has a value 1". If this condition is satisfied, that is if I1 equals three and I2 equals one, the result of the logical AND operation is one. This means that the arrow labelled one is followed to the next box. If I1 and I2 are not three and one respectively, the result of the AND operation is zero. This time the branching arrow labelled zero is used to define the next function. Let us assume that I1 equals three and I2 equals one. This will cause a branching from the top node to the termination box at the left which contains "State = 1". As mentioned before, this means that the search for the state of the subsystem is finished and the subsystem is in state 1.

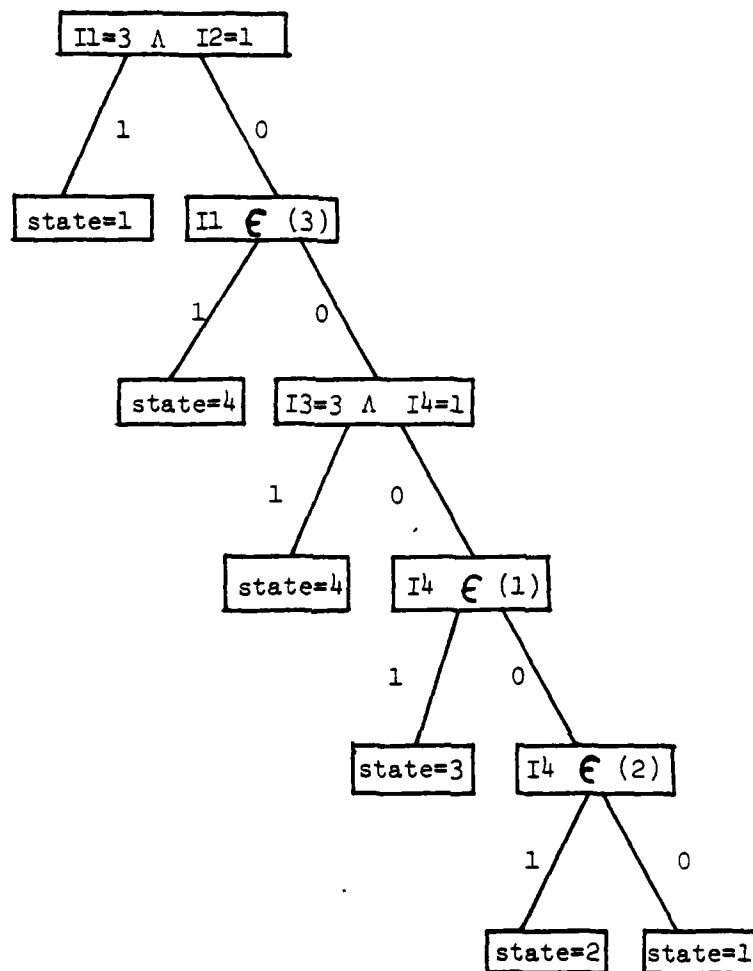


Figure 2-5. Example of Graphical Representation of Subsystems

If the first row of Table 3-2 is checked, it is seen that when  $I_1$  equals to three and  $I_2$  equals to one, the output is one. Here the values of the other inputs do not make any difference.

If the result of the first logical operation is zero in Figure 2-5 the next node is " $I_1 \in (3)$ ". Here if input  $I_1$  is three the result is a one, if not, the result is a zero. It branches to the termination box which assigns "state = 4" if the result of the logical operation is one, that is  $I_1$  equals three.

As seen from the above discussion drawing Figure 2-5 from Table 2-2 is a straightforward process. And Table 2-2 is formed from the table which given the states of the system for every combination of inputs.

Knowing the four inputs of this example subsystem, one can start from the top of Figure 2-5 and find the state of the subsystem by checking the conditions at the appropriate nodes. A computer program can work the same way. Starting from the top node and branching according to the results of the logical operations, the program will end up at a termination node where the state of the subsystem is given. The representation of subsystems by this method is suitable for input to computer routines. One such input format is developed and used in the following section.

The representations of the sequential subsystems are the same as the representations of the combinational subsystems. In sequential subsystems the present state of the subsystem is treated as another input to the subsystem. Even though there is no change in the representation, the computer program has to compute and record the states of the sequential subsystems.

All the subsystems in the network can be represented using this method. Then, given the values of the exogenous inputs to the system the states of all the subsystems are found. If the values of the inputs are changing over time, the network is used repeatedly to compute the states of the subsystem as a function of time.

As mentioned before the states of the subsystems in the network must be computed in a special order. At any given time the states of the subsystems that have the primitive variables as inputs must be computed first. These are the first level subsystems. Then given the states of the first level subsystems and the primitives, the states of the second level subsystems are computed. The computation proceeds in this order. To meet this condition the subsystems are sorted at the beginning of the execution of the program. This subroutine sorts the combinational subsystems into a tree structure and writes the sequential subsystems at the end of the list. At a given time  $t$ , given the values of the primitives, the states of the combinational subsystems are computed first. After the states of all combinational subsystems are known at time  $t$ , the states of sequential subsystems at time  $t + 1$  are computed. If there are any combinational subsystems that have inputs of sequential subsystems, their new states at time  $t + 1$  are computed next. The procedure of evaluating the states of the subsystems over time continues with the repetition of these steps.

## 2.2 Outline of the Program

A general computer program was written to compute the subsystem states of a network. It uses the techniques discussed in the previous section to represent subsystems. The input-output organization and logic of the routines are discussed in this section.

### Naming Subsystems

Subsystems are given two-digit integer "names". Numbers from 80 to 99 must be used for sequential subsystems. Any subsystem which is designated by a number between 80 to 99 will be treated as a sequential subsystem by the program. Here the primitives are treated as combinational subsystems. The primitive variables can be named by any number between 00 to 79.

### Input-Output Organization

There are three different types of inputs to the program: (1) the network, (2) structural representation of the subsystems, and (3) information about the exogenous inputs to the system.

The network used in a particular run is defined in an input file. An example of this file which represents the network graph in Figure 2-7 is shown in Figure 2-6. The network is established by listing each user defined subsystem and all subsystems which provide input to it. For example, subsystem 62 receives inputs from subsystems 04, 03 and 12.

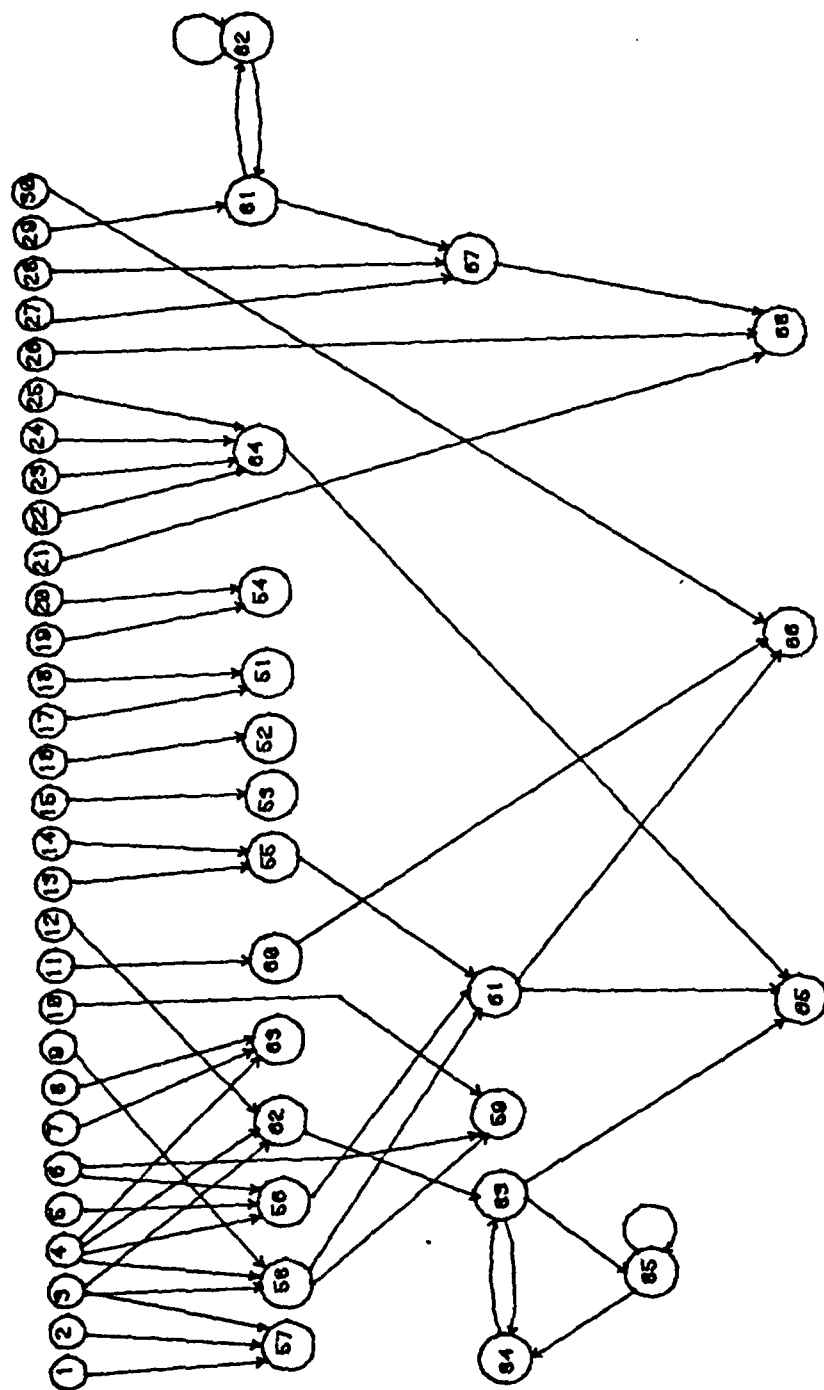
Each user defined (i.e. non-primitive) subsystem is given in a different record. There are 23 lines in Figure 2-6 which describe 23 subsystems. Each line is divided into 8 fields of two digits each. The first two-digit number in field 1 is the subsystem number. The following numbers in fields 2, 3, 4, 5, 6, and 7 are the numbers of the subsystems input to that subsystem. The two digit number in field 8 is accepted as the initial state of sequential subsystems. So the records representing combinational subsystems have blanks in field 8, but records of sequential subsystems must have their initial states in field 8.

The first line in Figure 2-6 shows that the inputs to subsystem 62 are the states of subsystem 4, subsystem 3 and subsystem 12. The third

62,04,03,12  
 64,22,23,24,25  
 65,63,64,57,61  
 61,56,55,58  
 60,11  
 63,04,07,08  
 55,13,14  
 66,59,60,30  
 56,04,05,06  
 57,01,02,03  
 58,03,04,09  
 59,08,58,10  
 67,27,28,81  
 81,29,82, , , , ,01  
 68,21,26  
 82,82,81, , , , ,01  
 83,62,84, , , , ,01  
 84,85, , , , , ,01  
 85,83,85, , , , ,01  
 51,18,17  
 52,16  
 53,15  
 54,19,20

Figure 2-6. Subsystems in the Network and the Network Representation





NETWORK REPRESENTATION OF EXAMPLE SYSTEM

Figure 2.7

line shows that inputs to subsystem 65 are states of subsystems 63, 64, 57, 61. The sixteenth line indicates that sequential subsystem 82 has an initial state of 01 and the inputs to this subsystem are subsystems 82 and 81.

Order is not important in this input file. A subroutine will sort this list into the proper precedence structure at the beginning of the execution of the program.

Structural representations of the subsystems are fed to the computer from a file with a format given by the example in Figure 2-8. This figure shows the structural representation of the subsystem in Figure 2-9 as an input to the programs. The inputs to the subsystem in Figure 2-9 are states of subsystems 63, 64, 57 and 61 which are represented by S63, S64, S57, and S61, respectively.

```
01,A,02,06,63,64,3,1
02,S,03,09,63, ,3
03,A,04,09,57,61,1,3
04,S,05,08,57, ,1
05,S,06,07,57, ,2
06,T,01
07,T,02
08,T,03
09,T,04
```

Figure 2-8. Example of Input File (2): Structural Representation Of the Example Subsystem

Each line in the input format corresponds to a node of the graph. The nine lines in Figure 2-8 represents nine nodes in Figure 2-9, (multiple copies of the state assignment nodes are counted only once).

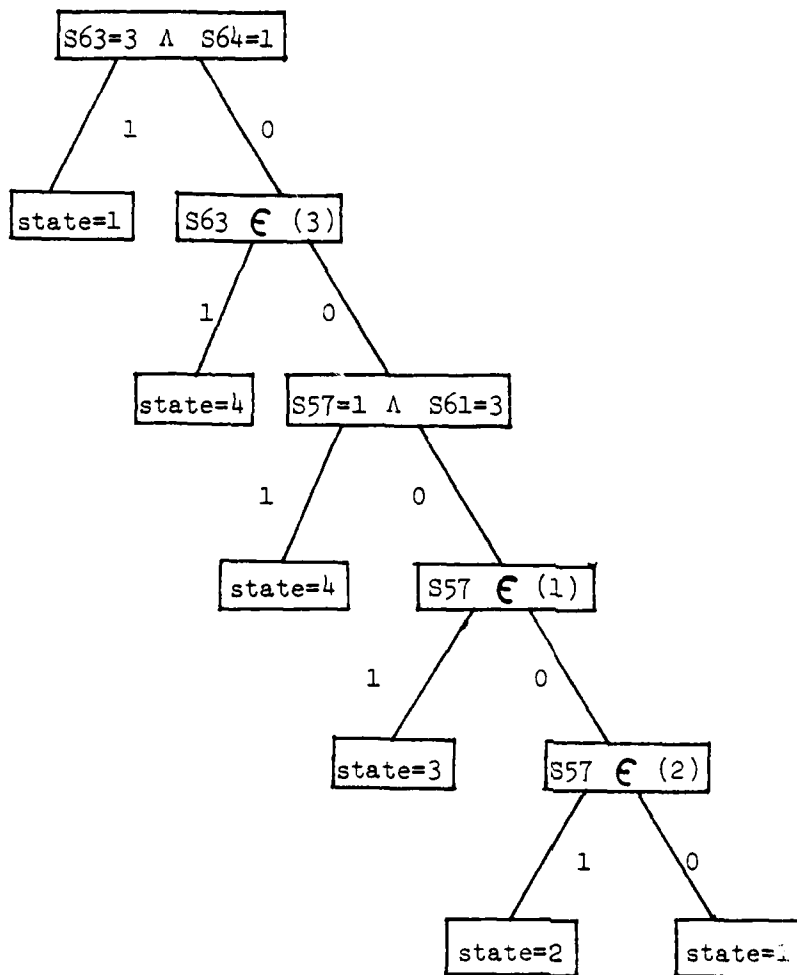


Figure 2-9. Graphical Representation of Example Subsystem

Records corresponding to termination nodes are given after all non-termination nodes are defined. The last 4 lines in Figure 2-8 define the 4 possible states of this subsystem.

Each line is divided into fields by commas. The first field has the node name (or box number from the graph representation of that subsystem) which is two digits. The second field has the symbol for the logical operation used in that node. Each operation is represented by one letter: "A" stands for "AND", "O" stands for "OR", "S" stands for "SET" and "T" stands for termination node.

If the record has a "T" in field 2, the third field represents the state to be assigned to the subsystem. The two-digit number in this field is the state of the subsystem if the computation ends with this record. If the record is representing a termination node, the fields following field 3 are not used.

If the record has one of the following letters "A", "O", "S", in field 2, it represents one of the logical operations AND, OR or SET. The third field gives the next node to be checked if the result of the logical operation is 0. The two-digit number in field 3 is the number of this node. The fourth field gives the next node to go to if the result of the logical operation is 1. Field 5 gives the first operand name as a two-digit number. Field 6 gives the second operand name as a two-digit number. In Figure 2-7, line one has "63" in field 5 and "64" in field 6. This means that operand 1 is the state of subsystem 63, operand 2 is the state of subsystem 64. Field 7 gives the value which operand 1 is to be compared to, field 8 gives the value the second operand is to be compared to.

There is a subroutine (SYOUT) which assigns the state of a subsystem given its representation in the proper format. The routine starts reading

the first line and searches down the input until a termination node is reached. Whenever a termination node is reached the search stops and the state of the subsystem is assigned.

All subsystems in the network, except the subsystems that are assumed to be primitives, are represented in this format in a separate input file.

#### Primitive Inputs To Data Analysis

The primitive variables of data analysis are contained in this input file. This file, which contains the time histories of these variables, must be in event format. Figure 2-10 shows an example of this file. The first two-digit number stands for the primitive variable number. The next five-digit number is the time at which the value was assigned, and the last one-digit number is the value assigned to that specific primitive variable at that time. The first two lines are header records. They are transferred literally at the beginning of the output file.

In this example the next thirty lines after the header record, define the initial conditions of the primitives, i.e. time 1 values. Since the file is event oriented only changing inputs, their new values and the time of the change are seen afterwards. For example there are five lines with time 2. This shows that only inputs 13, 14, 15, 19, 20 changed at time 2.

After time 2 there is not an event until time 126, when input 23 changes its value to 1. The next event is at time 134. This file must have the complete record of the input values in this form.

#### Output of the Program

The output of the program has the same format as the primitive input data explained in the previous section. Subsystems and their states are given in the output in event form.

1	22327	6
1	3 5101	
1	1	2
2	1	1
3	1	2
4	1	1
5	1	2
6	1	2
12	1	1
7	1	1
8	1	2
9	1	1
10	1	1
11	1	1
13	1	1
14	1	1
21	1	1
22	1	1
23	1	2
24	1	2
25	1	2
30	1	1
26	1	1
27	1	1
29	1	1
28	1	1
16	1	2
17	1	1
18	1	1
15	1	2
19	1	1
20	1	2
13	2	2
14	2	2
15	2	1
19	2	2
20	2	1
23	126	1
22	134	2
29	134	2
22	137	1
29	137	1
12	150	2
22	159	2
29	159	2
22	170	1

Figure 2-10. Primitive Variables

Figure 2-11 shows a part of an output file of the program. The first two lines are the header records copied from the input file. Then the initial states assigned to the subsystems are given.

The first 2 digits are the subsystem numbers. The time comes next. And the single digit at the end of each line is the state assigned to that subsystem at that time.

The third line in Figure 2-11 shows that subsystem 62 was in state 2 at time 1. The initial states assigned to other subsystems follow until the initial states of all the subsystems are assigned. Initial conditions are those assigned at time 1. After these there are four lines with time value 2. These lines show that the states of subsystems 83, 84, 53, and 54 changed at time 2.

After the initial states of the subsystems are assigned, the program reads the event base data about the primitive inputs. At each line it checks if that change of input affects any of the subsystems. If it does, it makes a list of those subsystems and assigns their new states. And it writes them on this file with the corresponding time.

#### Computer Programs

The computer program uses seven subroutines. The structure is given in Figure 2-12.

STATE is the main program. A rough flow chart of STATE is given in Figure 2-13. From the terminal it gets the names of the two input files: Input(1)-Subsystems in the network and the network structure, Input(2)-Primitive inputs. It also gets the name to be given to the output file which will store the states of the subsystems.

STATE then assigns the initial states of the subsystems and it reads the event base data of the primitive inputs and checks the combinational

1	22327	6
1	3 5101	
62	1	2
64	1	1
55	1	1
63	1	4
60	1	1
56	1	4
58	1	1
59	1	2
57	1	5
61	1	2
65	1	1
66	1	1
67	1	2
68	1	1
51	1	1
52	1	2
53	1	2
54	1	2
81	1	1
82	1	1
83	1	1
84	1	1
85	1	1
83	2	3
55	2	4
53	2	1
54	2	3
85	3	2
85	4	3
84	4	3
84	5	2
83	5	1
64	134	2
81	135	2
67	135	1
64	137	1
81	138	1
67	138	2
62	150	4
83	151	2
85	152	1
84	153	1

Figure 2-11. Subsystem States



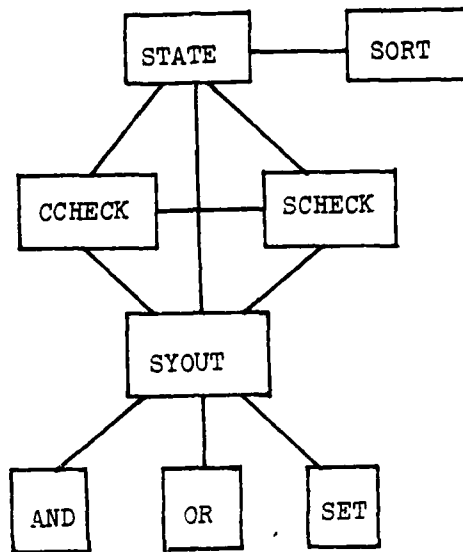


Figure 2-12. Structure of Routines

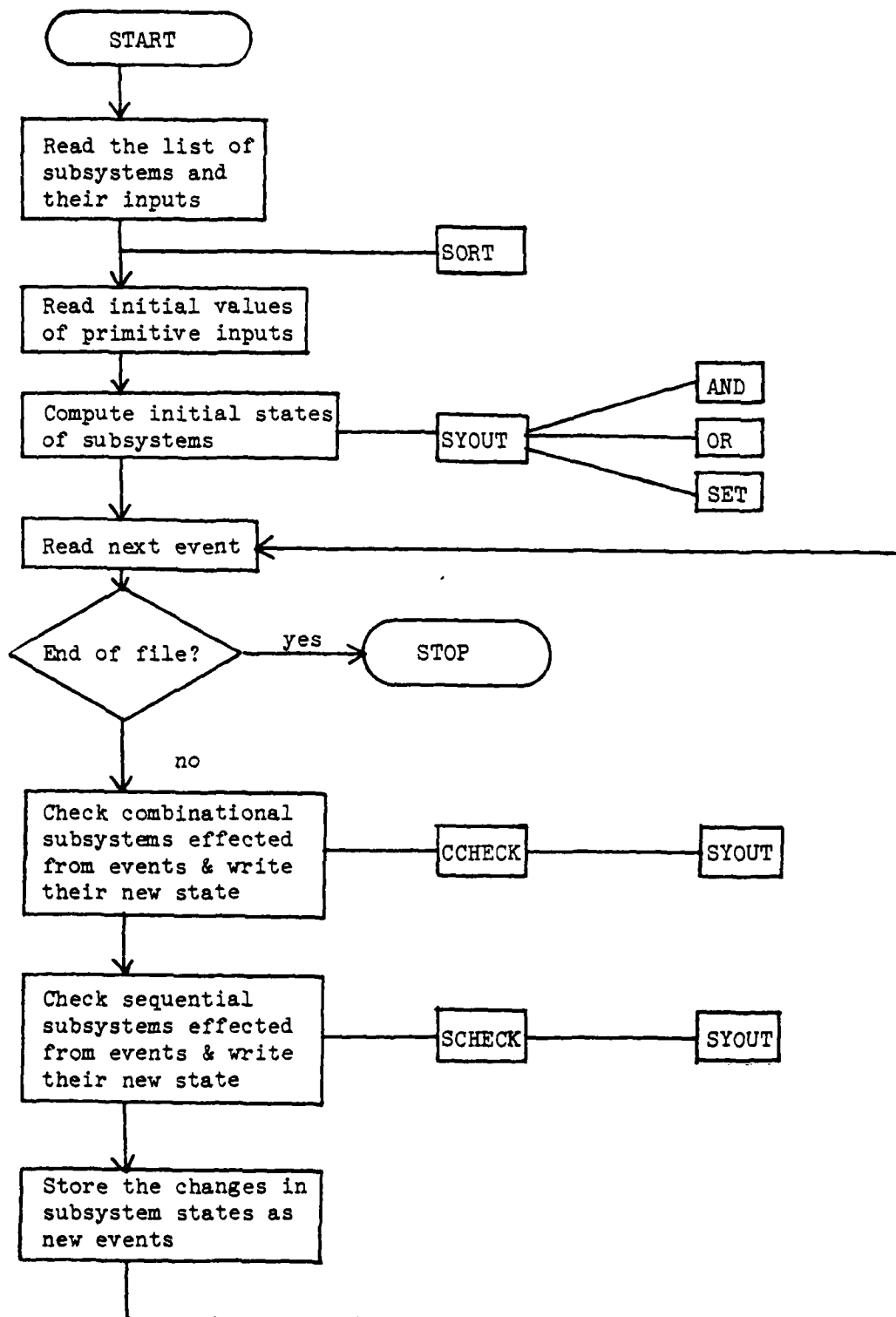


Figure 2-13. Flowchart of STATE

subsystems affected. Here it reads all the events which occurred at a given time before checking the subsystems affected. If there are any combinational subsystems affected by the events the new states are assigned and written by the subroutines CCHECK and SYOUT.

The sequential subsystems affected by the events are also found by SCHECK. Their new states are assigned by SYOUT. If the sequential subsystems oscillate, STATE finds their new states by using SCHECK and SYOUT over and over, increasing time by 1 each time. Meanwhile it also compares the time of the next event with the incremented time. If they are the same the next event plus the oscillating subsystems are considered to be the event set and the new state assignment continues. If the oscillating subsystems reach a steady state, the next change in the primitive inputs is read and the computation of states of subsystems continues.

At the beginning of the program, after the initial states of the combinational subsystems are computed, the sequential subsystems are checked for the initial condition affects.

Subroutine SORT is used to sort the subsystems at the first stage of the execution of the program. It was previously discussed that the combinational subsystems must form a tree. The states of all the subsystems which are input to that subsystem must be known before assigning its state.

This subroutine reads the subsystems and their inputs from input file (1). It sorts them such that every subsystem state is computable from the subsystems above it. It records the sequential subsystems at the end of the list.

This subroutine reads input file (1) which has the subsystem names and their inputs line by line. It checks the sequential subsystems and stores them in an array which will be added to the end of its output file. When it reads a combinational subsystem, it is checked if that subsystem is affected by any of the combinational subsystems below it. If there is a combinational subsystem below that is an input to it, the locations of the two subsystems are changed. Continuing this procedure all the combinational subsystems are sorted. The sequential subsystems are added to the list after the combinational subsystems. This list is recorded in a file called "ALLSYS.DAT". It has the same format as input file (1). ALLSYS.DAT is used by the main program instead of the input file (1).

The subroutine SYOUT reads the subsystem representations and assigns the states. It uses the subroutines OR, AND, SET. This subroutine is called with a specific subsystem. It then reads the input file of that specific subsystem line by line until its state is assigned.

Subroutine CCHECK finds the combinational subsystems affected by a set of events. It then writes their new states using SYOUT.

An event may change the state of a subsystem which talks to another subsystem. So the states of several subsystems may change in a chain. It keeps track of the changing subsystems and finds the others affected by them. Then it assigns the new states of all combinational subsystems that are affected by the event using subroutine SYOUT.

Subroutine SCHECK checks the sequential subsystems and finds the ones which are affected by the set of events. Events can be changes in the inputs to the network or changes in states of subsystems. Subroutine SYOUT is used to assign the new states of the changing sequential subsystems.

The sequential subsystems that change their states are returned to the main program as new events. Before writing the new states of the subsystems they are compared with their previous states. If they are the same, the state is not written again. This is required because there can be a change in one of the inputs to that subsystem which does not change its state. In other words two different values of an input to that subsystem may produce the same state.

Subroutines AND, OR, and SET perform the logical operations defined previously. Fortran code for this program and all subroutines can be obtained from the author. Further details are also available in Sevenler (1980).

### 3. DISCRETE CONTROL AND THE DESIGN OF INTEGRATED DISPLAYS

This section is a brief review of the Ph.D. dissertation of Christine Mitchell which utilized discrete control concepts in the design of a computer integrated display. Details not presented here are available in Mitchell (1980) and Mitchell and Miller (1980).

#### 3.1 Background and Rationale

In selecting and constructing a system which could be used both to illustrate the utility of the discrete control modelling methods in display design and to experimentally test the resulting displays, several considerations were taken into account. The system had to be a reasonable, although somewhat idealized, representation of a control task faced by a real controller in an information intensive control situation. The control system had to allow multiple control activities and had to require multiple pieces of displayed information in order to make most control decisions, and yet be simple enough to allow subjects participating in the experiment to function as trained controllers within a reasonable length of time. Finally, control performance had to be quantifiable so that meaningful comparisons of operator performance under varying display conditions could be made.

A discrete control routing task fit these requirements. Although a continuous control task (e.g., tracking) could potentially be modelled using the discrete control methods by discretizing the control outputs, the development of a discrete control model is much simpler for naturally discrete systems. Furthermore, a discrete control system is conceptually simpler and requires less motor skill training for participants than most continuous tasks.

In addition to modelling and experimental expediency, an idealized, discrete control task has a number of substantive advantages. Modern systems are increasingly automated with the result that the human operator has been elevated from the position of manual controller to that of a supervisor and monitor. The human supervisor of a complex system performs upper level, goal oriented functions such as planning system activities, programming the computer, monitoring the system behavior, adjusting on-line parameters, and intervening to take over direct control in abnormal or emergency situations (glossary in Sheridan and Johanssen, 1976). Roscoe and Eisle (1976), for example, described the pilot of the new generation of high-speed, multimission aircraft as an "information manager or a fast decision maker as opposed to a direct controller of flight variables". The aperiodic, event based nature of the supervisory function makes it a likely candidate for a discrete, event based model. The routing system used in this investigation requires control activities which are representative of the monitoring, system tuning, and intervening behaviors of a system supervisor.

Singleton (1976) in distinguishing models of the supervisor from the traditional models of the controller suggests that the former are more complex, requiring explicit modelling of at least two additional dimensions not required in the simpler models of the manual controller. Models of the supervisor require a dimension which captures patterns of events in time and space which precede control activities as well as the inclusion of a dimensions which models the purpose of a control activity, described perhaps at multiple levels. Singleton characterizes the traditional models as stimulus-response models and describes the enhanced models as gestalt or cognitive, capturing environmental conditions and operator purpose.

A distinct advantage in the use of the discrete control modelling methods to guide display design is that the methodology deliberately attempts to model the high level operator functions or purposes which govern individual control activities; in addition, the next state transition functions provide a mechanism for modelling event sequences which cause or influence system changes and operator actions.

Thus, although the specific system used in this investigation is a manual or discrete control task, the discrete nature of the control actions and the slow dynamics of the system allow a generalization of the results to more supervisory control situations.

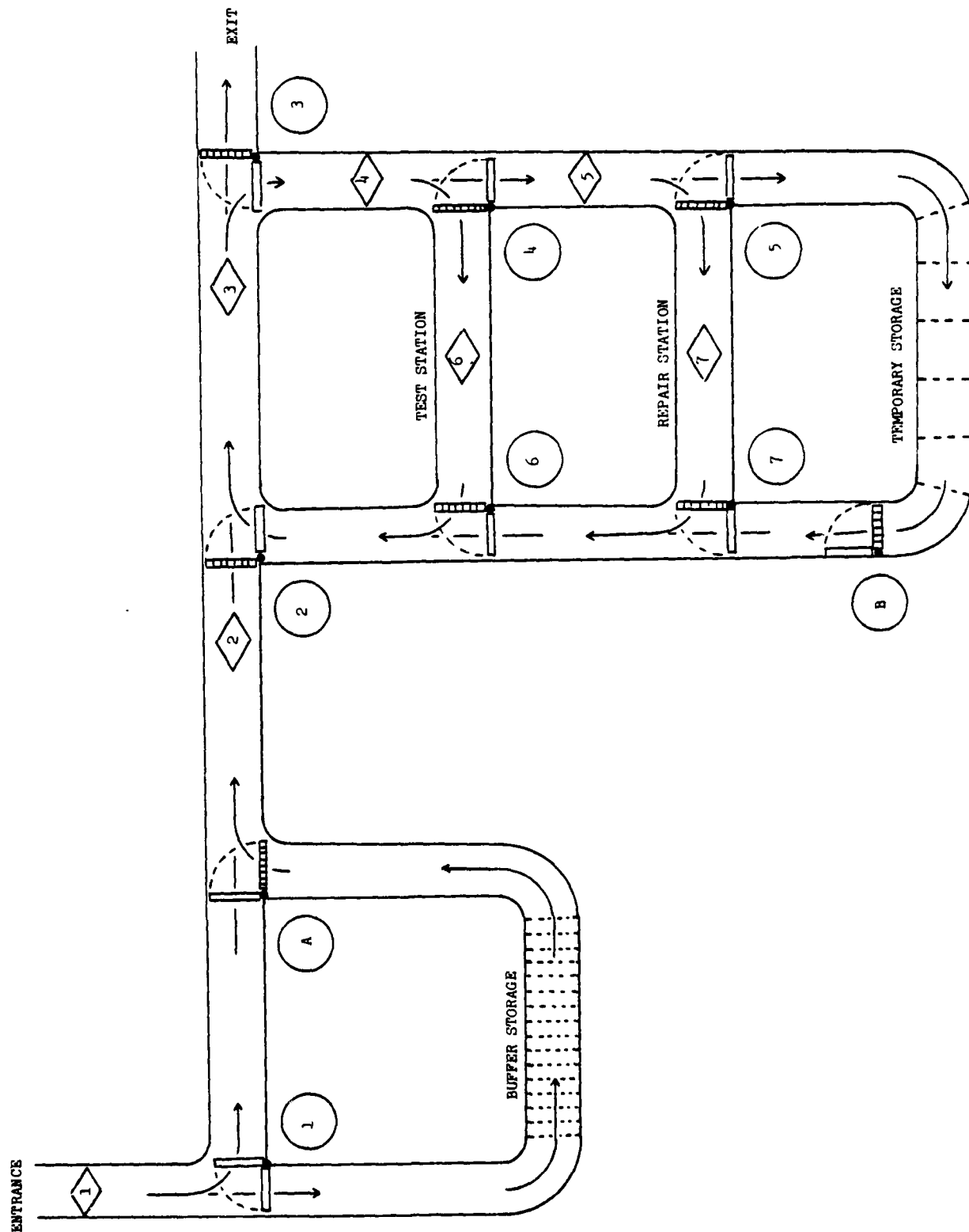
### 3.2 System Dynamics, Controls, and Constraints

The system chosen for this research is a conveyor network which is used to route engines through various checkpoints. In general, engines are routed from the entrance, to the testing area, to the repair area, if necessary, and, when successfully tested, out of the system. Figure 3.1 is a schematic of the system.

The system components consist of seven stations, three conveyor tracks, and two storage areas. The stations are locations equipped with sensors where information about arriving or waiting engines is conveyed to the operator. Space limits the number of engines which can be held at a station to at most one. The conveyors are unidirectional tracks which are pressure sensitive. A track moves only when an engine is loaded onto it; the conveyor carries the engine to the next station, stopping when the engine arrives at the station. Once an engine is moving on a conveyor there is no information available about the engine until it arrives at the next station. Engines can only be loaded onto nonmoving tracks; thus, the maximum number of engines allowed on a track is one.



Figure 3.1  
Schematic of the System



The storage areas are FIFO queues in which engines can be held indefinitely. The Buffer Storage area has no capacity limitations; Temporary Storage has room for no more than six engines.

The system is controlled by means of a set of nine switches. The switches labelled 2 through 7, A and B, control a mechanism which acts like a gate with springs. The gate has a natural or resting state, denoted by the hash marks in Figure 3.1; the gate remains in the resting state until the operator pushes a control switch. When the corresponding switch is depressed the gate momentarily opens, long enough to allow a waiting engine to roll past; after a few seconds the gate automatically closes, reverting to the resting state. Switch A controls the gate which, in the resting state, blocks the exit of the Buffer Storage area; when switch A is depressed the gate opens allowing the first engine in the Buffer Storage queue to roll out onto the Production Buffer Transportation Track. Switch 2 controls the gate at Station 2; when the switch is depressed the waiting engine rolls out onto the Test-Repair conveyor. Switches 6, 7, and B operate in a similar manner. The closed gate prevents the waiting engine from moving; when the switch is depressed the gate swings open allowing the waiting engine to roll out onto the conveyor. After a few seconds, the gate automatically reverts to the resting state.

Switches 3, 4, and 5 have a slightly different function in that there are two paths out of the associated stations. As a general rule, an engine arriving at a station will pause for a few seconds; if no operator action occurs at Stations 3, 4, or 5 within a short period of time the waiting engine will continue moving on the default path. At Station 3 a waiting engine will default down to the Feed Track moving toward Station 4. Unless prevented by operator action, an engine arriving at Station 4 will

pause for a few seconds and proceed down to Station 5. In a similar manner, an engine arriving at Station 5 will pause for a few seconds and, unless the operator prevents it, will continue down to Temporary Storage. If switch 3 is depressed with a successfully tested engine waiting at Station 3, the engine is routed out of the system. If switch 4 is depressed, the engine waiting at Station 4 is routed into Station 6. If switch 5 is depressed, the engine waiting at Station 5 is routed into Station 7.

Switch 1 controls a different sort of mechanism; the gate at Station 1 has no automatic response. The state of the gate must be manually set; it remains in a given state until it is manually reset. When an engine arrives at Station 1 it will pause for a few seconds. If there is not operator action at switch 1, the engine will proceed to Buffer Storage if the gate is currently closed to Station 2; or, if the gate is opened to Station 2, the engine will default onto the Production Buffer Transportation Track, moving toward Station 2. If the operator changes the state of the gate while an engine is waiting at Station 1 the engine will immediately proceed in the direction defined by the new gate state.

Some stations can be used as holding areas for single engines. An engine arriving at Station 2 will remain there indefinitely, leaving only when switch 2 is pressed. Likewise, engines with repairs or testing completed will wait indefinitely in Stations 6 and 7 until the respective switch is depressed. Engines can also be held at Stations 4 and 5, though this procedure requires some additional operator action.

The remaining system controls available to the operator are the hold mechanisms which will hold engines indefinitely at Stations 4 and 5. The

hold switches are both two state devices. To hold an engine the switch must be depressed once; to disengage the hold mechanism, the switch must be depressed again. As there is no mechanism for holding engines at Stations 1 and 3, engines may never be held at either station.

Completed engines can be rerouted back into the system and can be sent into the Test or Repair Stations. More generally, engines not requiring test can be routed into the Test Station and engines not requiring repair can be routed into the Repair Station. However, there is a time penalty for such actions in that these engines are held in the stations for a period of time before they can be routed out. Those periods in which an engine is trapped in the Test or Repair Station are referred to as local station locks. The engine can not be routed out until the station is unlocked; a state change which is system as opposed to operator controlled.

The Test-Repair Division Routing System processes two types of engines. At any given time the operator is asked to give processing priority to one engine type or the other. Although throughput is always a goal the operator is asked to give priority to the specified engine type whenever possible. Operator performance is evaluated by an algorithm which takes into account throughput, number of expedited engines processed as well as the number and type of errors.

These system components, dynamics, and constraints constitute the Test-Repair Routing System. The discrete control model described in Mitchell (1980) formalizes these entities and relationships in a form particularly suited to assisting in the design of information displays for system control.

### 3.3 The Experimental Conditions

This experiment consisted of three experimental conditions.

Integrated information displays, the Grouped Primitive and Hierarchic/Preview displays, constituted two of the conditions. The third was a display configuration which simulated a dedicated information display scheme. The dedicated information display was designed so that the status of each of the system components was given on the operator's console. The information was presented in parallel and the operator had no control over what information was displayed. The control panel for this display condition consisted only of system controls. A change in system state caused the corresponding displayed information to be updated. There was no additional mechanism (e.g., reverse video, blinking), however, to alert the operator to a state change. The operator was required to carefully monitor the displayed information in order to detect changes.

The dedicated display did not provide any diagnostic information to assist the operator in correcting a locked condition. When a system lock occurred, the operator was alerted to the fact by a flashing message; however, no additional information concerning location or corrective procedures was forthcoming. In addition, the dedicated display provided no preview information alerting the operator to upcoming events.

Error messages for all three display configurations were the same. When the operator made an error, an alarm sounded and a description of the error was displayed on the console. The operator was told either that the system had overridden the control action, thus the operator was able to proceed to another activity, or that the system was locked and further operator action was required to unlock the system.

The attributes which characterize the three displays used in this experiment are summarized in Table 3.1. The primary difference is in the control of information. The contents of the two integrated display configurations were primarily operator controlled; whereas, the dedicated display permitted no operator control of the screen contents. The dedicated display was parallel in form; the integrated displays were serial over control functions and parallel within control functions. The dedicated display had a dynamic update capability. The integrated displays provide static snapshots of the system; to update the display contents, an information query button had to be depressed again. The Hierarchic/Preview display was the only display provided the operator with preview information which alerted him or her to forthcoming events. Both of the integrated displays contained diagnostic information to assist the controller in correcting system locks. These displays also provided system alerts, notifying the controller of the occurrence of events likely to require operator action. The dedicated display condition provided neither diagnostic information nor system alerts. Finally, both the dedicated and Grouped Primitive displays were composed of low level pieces of data directly linked to hardware components. The Hierarchic/Preview display, on the other hand, provided information which was aggregated and reformulated depending on system state.

#### 3.4 The Control Station and Information Display Console

The control station for the Test-Repair Division Routing System is composed of a control panel and a display console. The control panel consisted of a schematic of the routing system with the system controls located adjacent to the corresponding gates or stations. Two panels were used in this experiment. The panel for the dedicated display condition

TABLE 3.1

## Comparison of Display Attributes

Dedicated Display	Grouped Primitive Integrated Display	Hierarchic/Preview Integrated Display
System Controlled	Operator Controlled	Operator Controlled
Parallel	Serial	Serial
Dynamic Update	Static Snapshot	Static Snapshot
No Preview Information	No Preview Information	Preview Information
No Diagnostic Information	Diagnostic Information	Diagnostic Information
No System Alerts	System Alerts	System Alerts
Low Level Data	Low Level Data	Reformulated High Level Data

contained only the control buttons. For the integrated display conditions, display controls were added to the system controls on the second panel; the information query buttons for control functions were placed near the system controls to which the functions correspond. The same panel was used for both integrated display configurations. As the only physical difference between the two was the inclusion of the "Shift" button for the Hierarchic/Preview display, the panel was constructed so that this button could be easily removed and the hole covered.

The information display console was a 19" video monitor located at eye level for a seated subject. The information displayed on the console, for all three display conditions, was computer-generated text material. The video monitor was linked to the computer via a remote jack on a Regent 100 CRT terminal. The Regent terminal served as the experimenter's terminal and was not used by the subject while controlling the system.

Information displayed on the console for the dedicated condition was automatically updated at the occurrence of any event. Operator controlled information for the two integrated displays, however, consisted of snapshots of the system. When information for a particular control function was requested the current status of the relevant systems was displayed on the screen; the displayed states, however, were not automatically updated. An information update could only be acquired by once again requesting information. As information was dated as soon as it was displayed, the operator was encouraged to frequently sample the console. Requested information was rapidly displayed, typically requiring less than three seconds to completely display all the required information. Information was changed or updated by depressing information query buttons. Displayed information was deleted whenever a control action was performed. A control



action replaced the previously displayed information with an interim message, indicating that more information was available upon request.

For the integrated information displays, error messages were displayed on the top several lines of the display. The brief, system controlled messages alerting the operator to events potentially requiring operator response were displayed on the right side, a bit above center. The continuously displayed expedited status was directly opposite, on the left side of the screen. The remainder of the screen was devoted to the display of operator controlled information. The screen format for the dedicated display condition also had the error messages displayed at the top of the console; the remaining portion of the screen was filled with the status of the individual system components.

Error messages remained on the screen for a fixed, ten second period or until the next operator action. With the exception of test and repair completions, system controlled messages, displayed in reverse video in order to be more noticeable, also remained for a ten second period or until the operator initiated an action which responded to the message. Messages notifying the operator of test or repair completions remained on the screen until the waiting engines were released.

### 3.5 Data, Performance Measures, and Incentives

Experimental data were collected by the PDP 11/34 minicomputer which generated the simulation and information displays. The data consisted of a log recording the time and event code of both system and operator initiated events. Changes in the system components caused by exogenous inputs, system dynamics, and operator actions as well as information query requests and operator errors constituted this discrete event data base.

At the conclusion of each experimental session, an analysis program was run to provide summary information on the subject's performance.

The performance measures consisted of an error count, a throughput count, and an operator score which aggregated and weighted errors, completed engines, and expedited engines. The operator score was a measure which was reported to the subjects; it was a measure that the subject's were directed to optimize. Operator errors were divided into two types. The first type, costing the subject one point each in the computation of the session score, consisted of more serious errors. Such errors included operator actions which locked the system, failed to unlock a locked system, or directly violated a system constraint causing a safety system override. The second type of error was more minor, costing the subject only a half point each. These error typically occurred when an operator needlessly activated the controls (e.g., depressing Switch 2 with Station 2 clear). Subjects were penalized for such actions in order to encourage careful, thoughtful control actions supported by necessary information gathering, and to discourage random button slapping. Information was free and the system was comparatively slow, allowing adequate time for information gathering. Completed engines which were routed out of the system at Station 3 earned the subject two points each; if, in addition, the departing engine was of the type currently expedited, the operator received a third point.

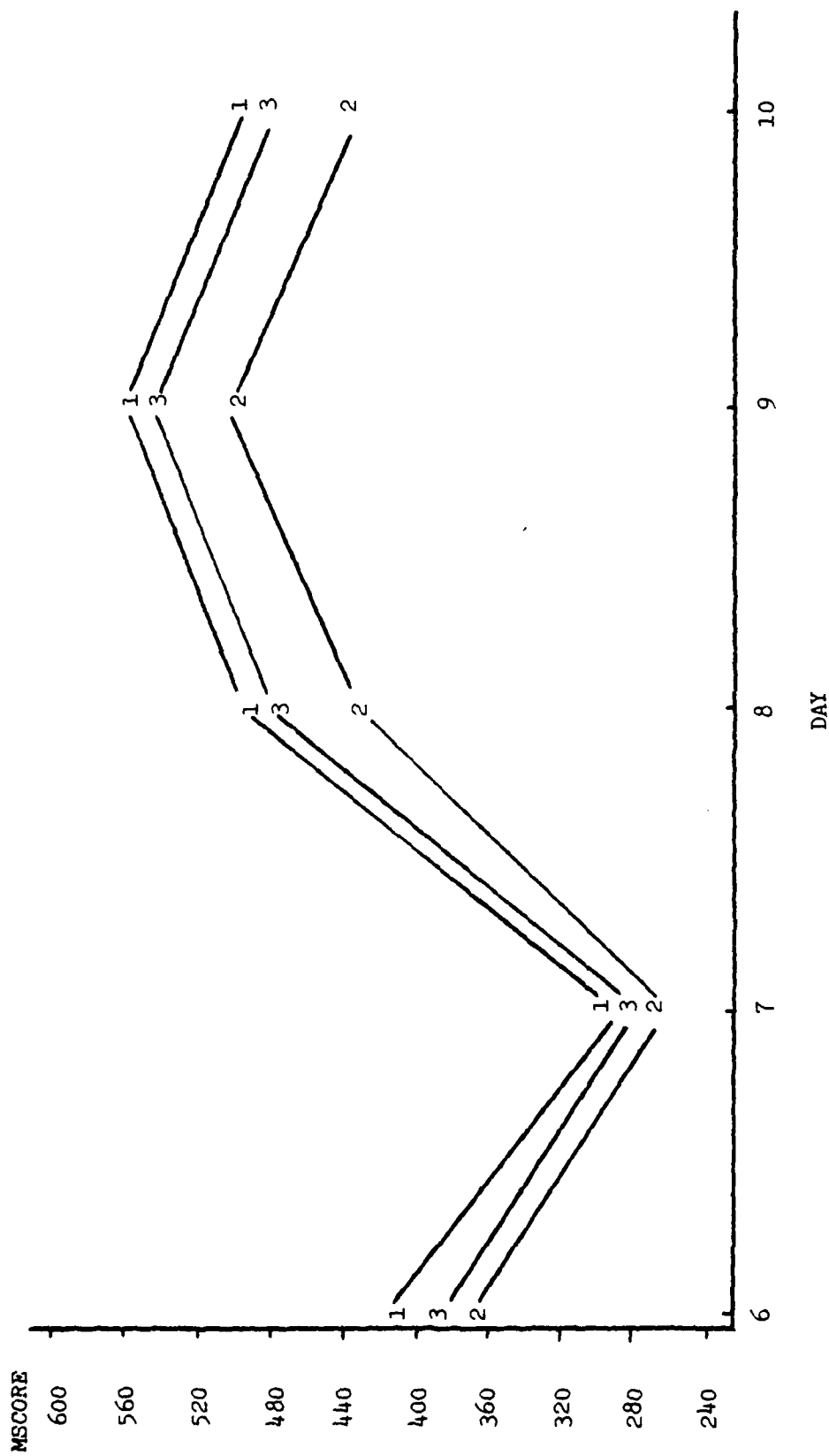
As the purpose of the experiment was to ascertain the effect of the displays on operator performance, trained performance, not reflecting learning effects, was desired. In order to eliminate, or at least mitigate the effects of learning, only the data of the last several

sessions was used for analysis. Participants were told that the first sessions were training sessions and that those sessions which constituted the usable data would only be specified at the conclusion of the experiment.

### 3.6 Results

The naming convention used in the plots and throughout this discussion represent the dedicated display as display condition one, the Grouped Primitive display as display condition two, and the Hierarchic/Preview display as display condition three. For each type of plot, six dependent measures were calculated: session score (SCORE), number of errors per session (ERRORS), number of engines processed (ENGINES), number of expedited engines processed (EXPED), the percentage of time that Station 6 was engaged in testing engines (BUSY), and the percentage of time that Station 6 held an engine (HELD). In order to distinguish between display conditions, data for display condition one was plotted against day + .2 and data for display condition three was plotted against day + .4.

The mean value of scores plotted over days suggest some extremely nonrandom behavior (Figure 3.2). For a given day, the random number streams defining system inputs and exogenous events on that day account for a great deal of the day to day variation in session score. On day seven, for example, a very high percentage of tested engines failed resulting in comparatively lower session scores for all subjects under all display conditions. The graphs clearly show, however, the differential effects of the displays over the five days. For all days, the mean score of subjects who controlled the system under display condition one was superior to that of subjects who controlled the system under display condition three, which in turn was superior to the mean score for subjects



MEAN SCORE PER SESSION BY DISPLAY TYPE

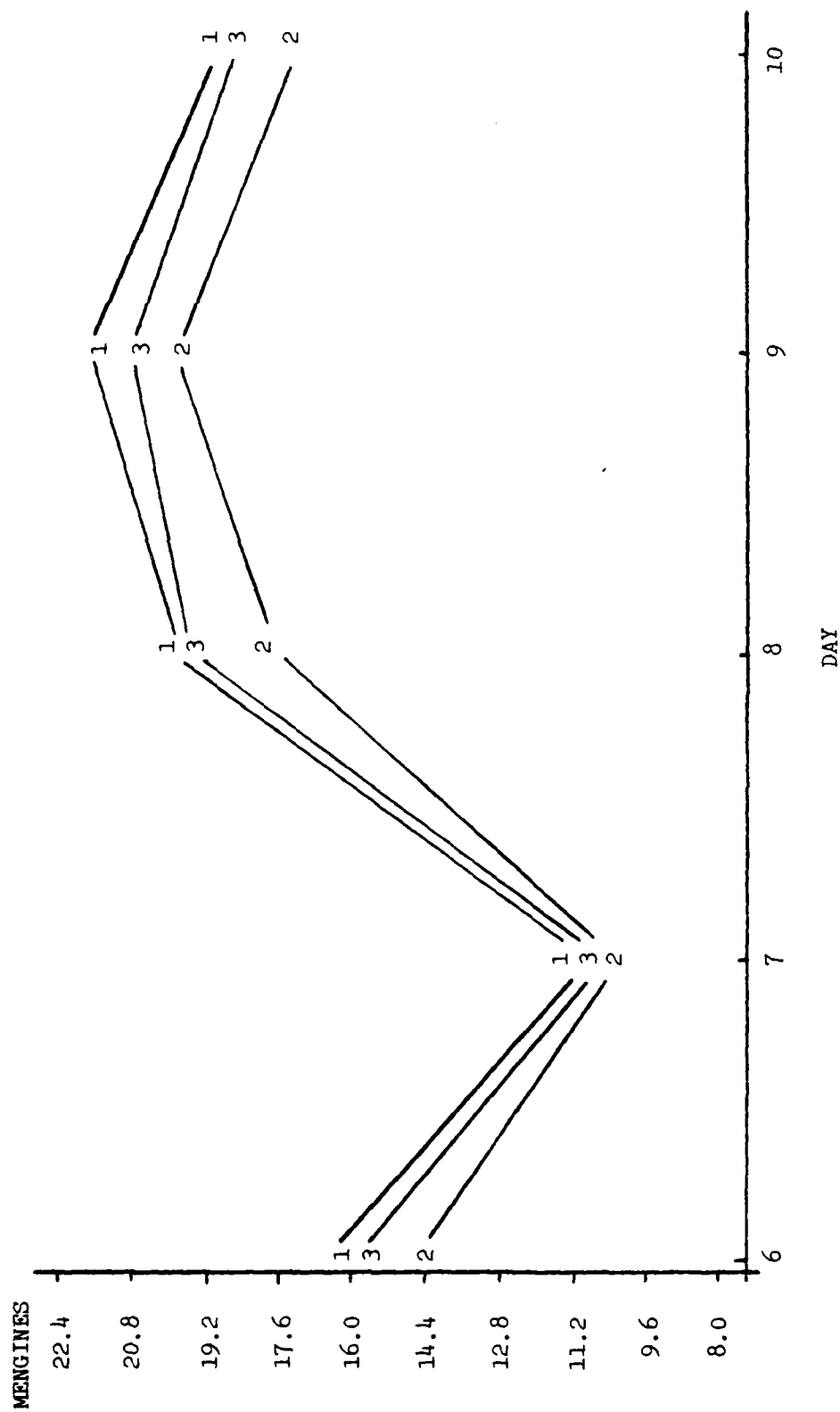
FIGURE 3.2

controlling the system under display condition two. On average, the difference between displays three and two seems to be twice as large as that between displays one and three.

This pattern is repeated in the graph of mean number of engines processed per day (Figure 3.3). The mean for display condition three closely tracks that of display one. The mean number of engines processed with display two consistently lags behind; with the difference between conditions three and two twice the size of the difference between three and one.

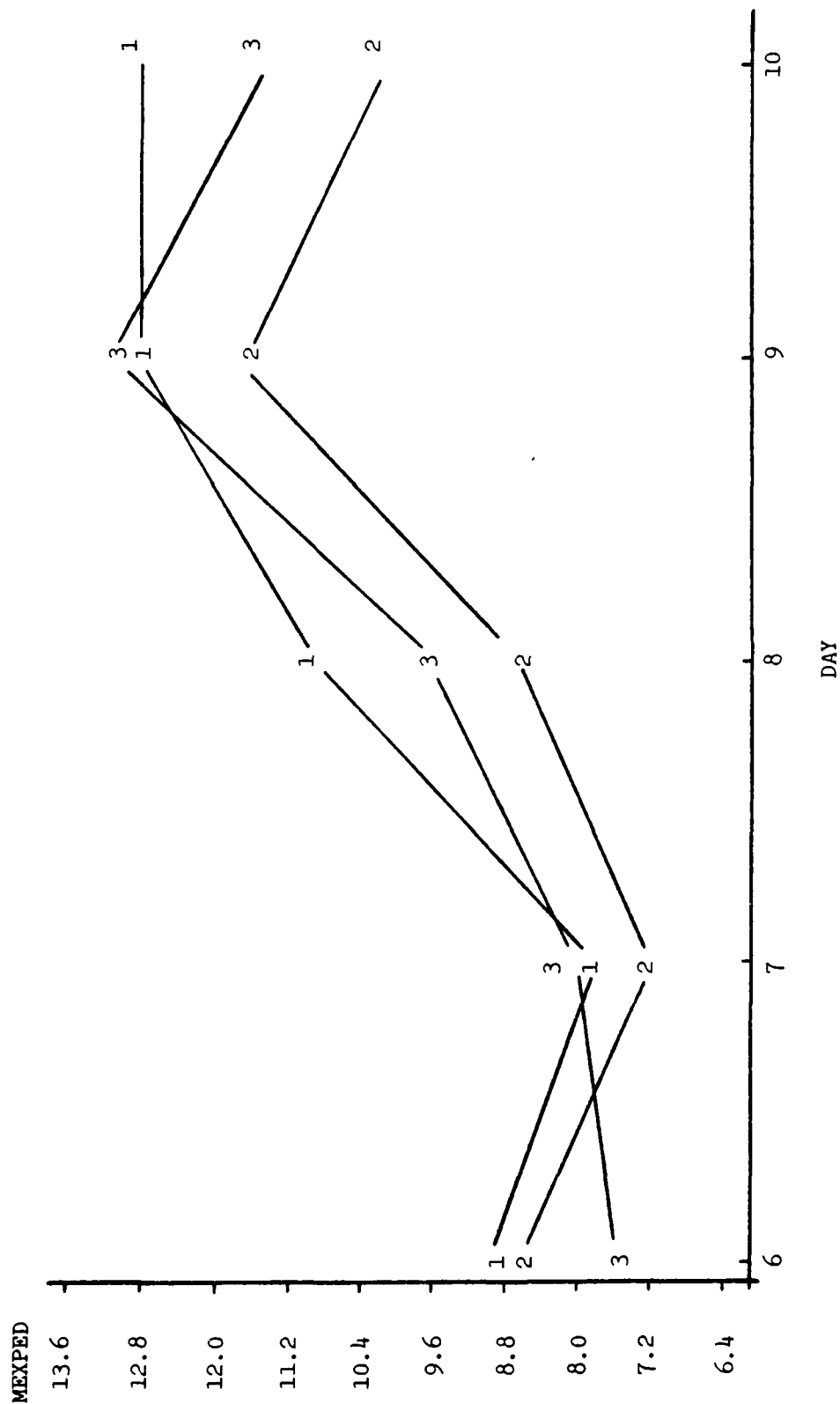
The graph of the mean number of expedited engines processed shows a slightly different pattern (Figure 3.4). The relation between display conditions one and two is consistent with the previously observed pattern. There is, however, more variability in the performance under display condition three. For day six, the mean number of engines expedited is lowest for display condition three. For days seven and nine the mean number of engines is slightly higher for display three than for the other two display conditions; the remaining days follow the previously noted pattern in which performance under condition one is superior to that under condition three which was superior to that using display condition two.

The graph for mean error demonstrates relationships between displays which are similar to those for expedited engines (Figure 3.5). Performance under display condition one is uniformly superior to that under display condition two. The error pattern for display condition three shows more variability. Display condition three had the fewest mean errors for days six and ten, and the greatest number of errors for days seven and nine. It should be noted, however, that the overall differences in numbers of errors made for the different displays is minimal; the mean number of errors is under two per day, regardless of display condition.



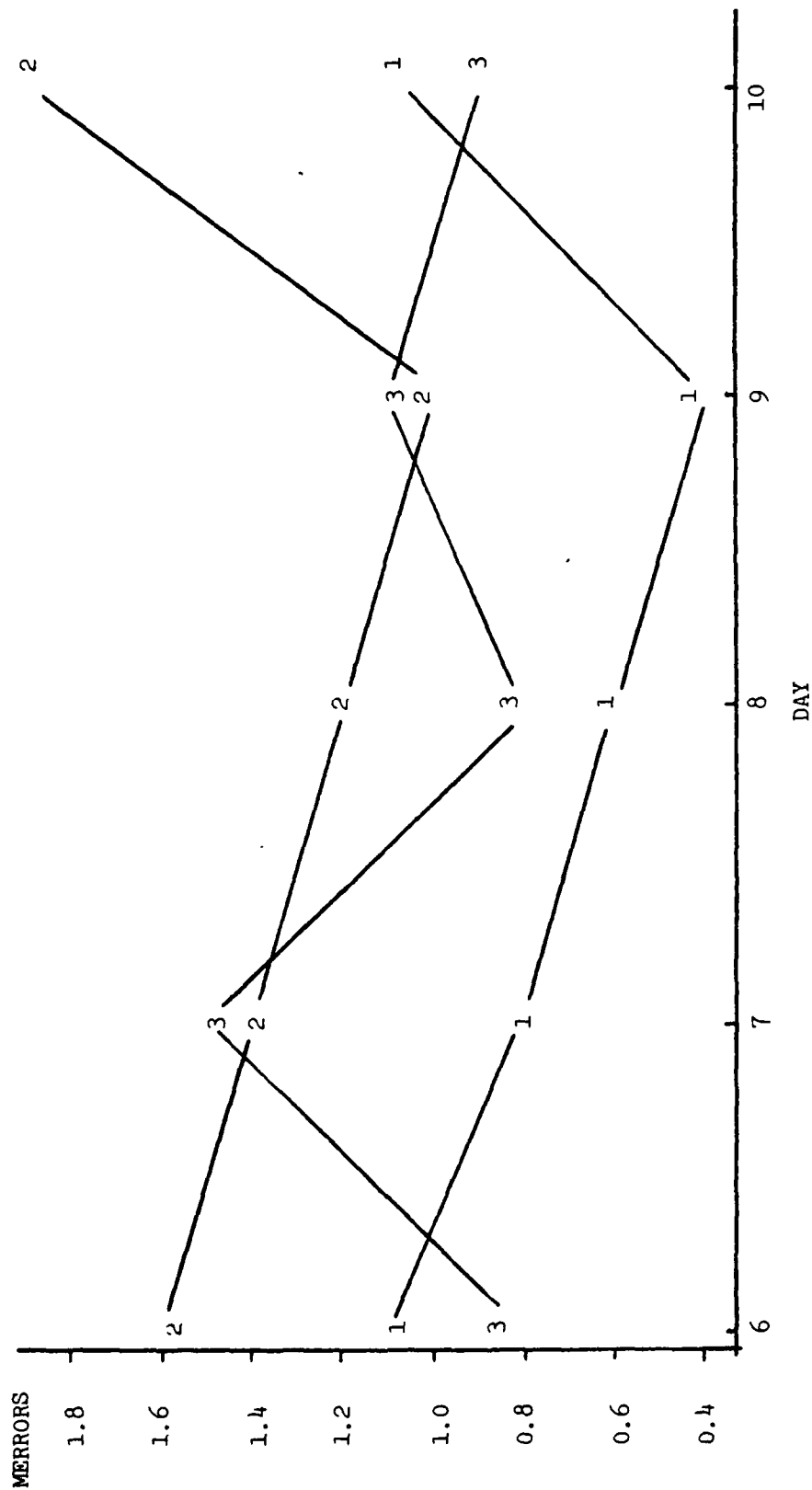
MEAN NUMBER OF ENGINES PROCESSED PER DAY BY DISPLAY TYPE

FIGURE 3.3



MEAN NUMBER OF EXPEDITED ENGINES PROCESSED PER DAY BY DISPLAY TYPE

FIGURE 3.4



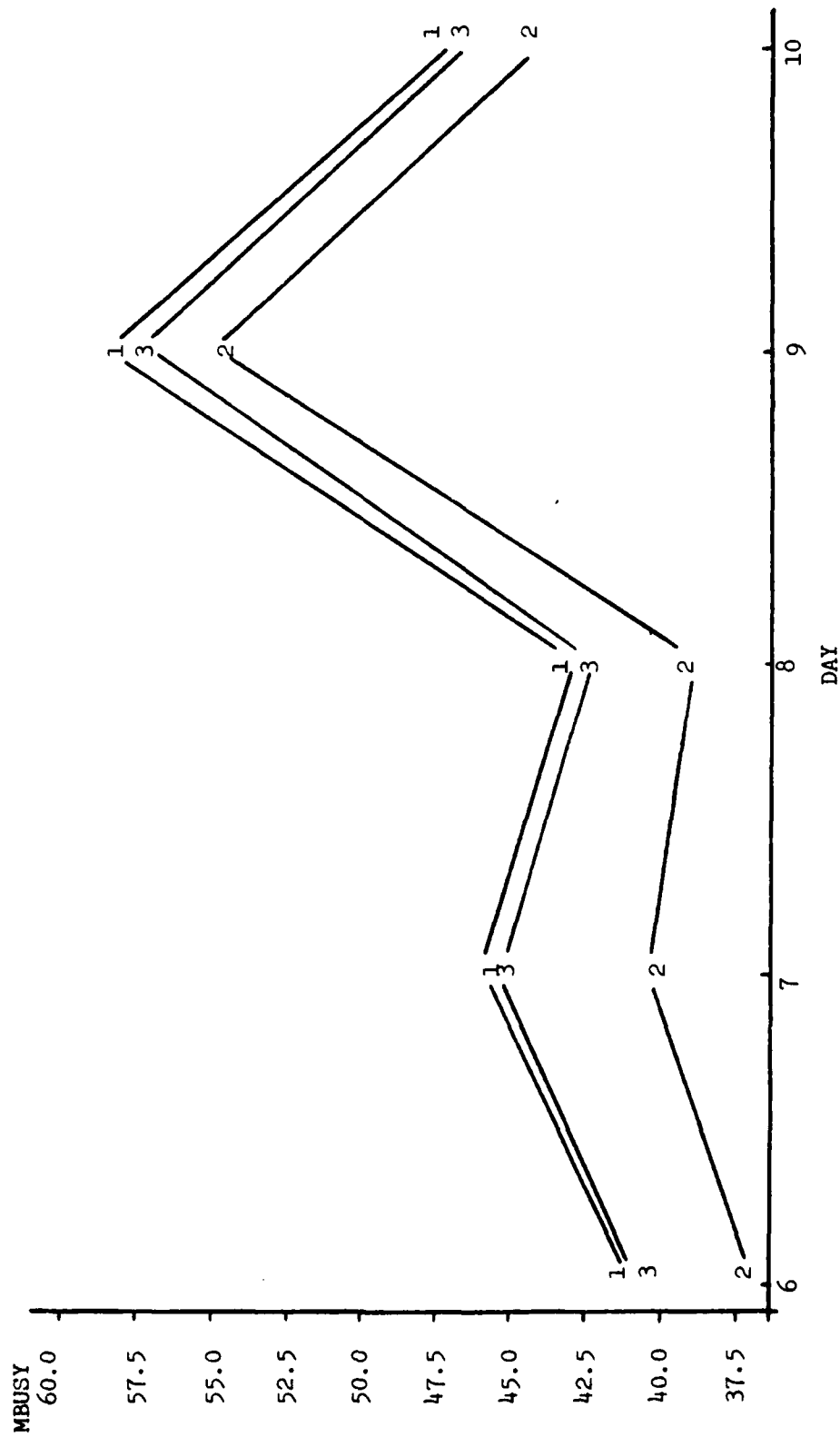
MEAN NUMBER OF ERRORS PER DAY BY DISPLAY TYPE

FIGURE 3.5



The last two measures used to investigate potential differences in operator performance attributable to displays are resource utilization measures for Station 6, the Test Station. Some indicator of operator efficiency was desired. Error rate is essentially a sufficiency measure, used to put a lower bound on the operator's understanding and skill. The error rate measure has the advantage of being free from the effect of the randomly generated, exogenous events driving the simulation on the respective days. A measure of operator efficiency, free of the effects of day to day variability, was sought. It was hoped that the Station 6 utilization measures, percentage of time that the station was engaged in testing (BUSY) and the percentage of time that the station held an engine (HELD), would be such measures. The data indicates that BUSY reflects a good deal of the system induced variation; HELD, however, appears fairly stable across days. The HELD measure is an indicator of strategy and the positive correlation between HELD and score suggests that a strategy which keeps Station 6 full is a successful strategy. With this type of strategy, the operator puts a very high priority on keeping an engine waiting at Station 4 for entry into Test; generally, an engine is only released from Station 4 when a second engine is awaiting entrance at Station 4. This policy dictates operator actions for releasing engines from Station 6, routing engines from Station 2, Station 7, and Temporary Storage. Keeping Station 6 full was neither encouraged or discouraged in training subjects; the subject's ability to note the importance of this was an indicator of her or his strategic aptitude.

Examination of Figure 3.6 shows that the mean value of BUSY exhibits the same ordering across display conditions observed in score and engines processed. Except for day six, mean BUSY was greatest for subjects using



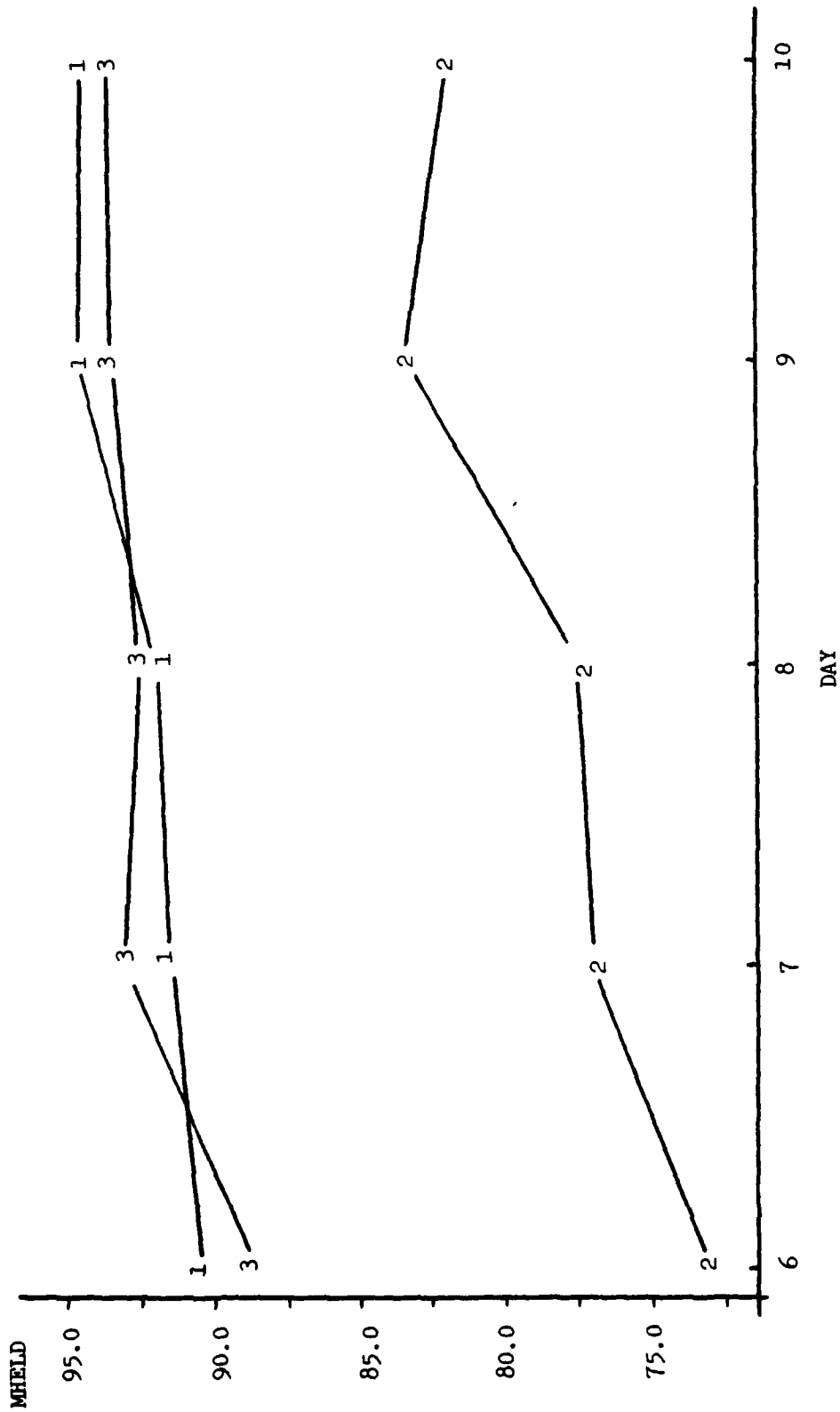
MEAN PERCENTAGE OF TIME AN ENGINE WAS ENGAGED IN TESTING  
PER DAY BY DISPLAY TYPE

FIGURE 3.6

display one, and lowest for display two; performance with display three was in the middle, closely tracking condition one. On day six, mean performance with display three was marginally higher than that for the other two.

The HELD measure shows an upward drift for mean values (Figure 3.7). This is particularly true for display condition two. As this measure is an indicator of strategy, the drift may imply a lack of a fixed strategy. Once again, display conditions one and two follow the same pattern of variation; performance under display condition one is consistently superior to that under display condition two, measured by both mean and median values of HELD. Performance under display condition three is also superior to display condition two. The relation between performance under display conditions three and one shows no consistent pattern, the close values and lack of a consistent ordering between them may suggest that the difference is merely random variation.

Taken together, the data suggest several things. First, the randomly generated variation in the system has more effect on day to day performance, for the majority of measures, than the different displays. Second, performance with the three displays seems to differ. Display conditions one and two differ greatly and consistently on all of the performance measures used in this study. Performance differences appear to exist between displays three and two, though not as consistently or strongly as between one and two. Overall, display condition one seems to result in performance that is, on average, slightly superior to that using display three; whereas, both of these displays results in performance which, for the most part, is superior to that of display two.



MEAN PERCENTAGE OF TIME AN ENGINE WAS HELD IN STATION 6  
PER DAY BY DISPLAY TYPE

FIGURE 3.7

### 3.7 Discussion of the Results

The basic conclusion is that subjects who controlled the system using the dedicated information display or the Hierarchic/Preview information display, on average, performed significantly better than those who controlled the system using the Grouped Primitive display. Furthermore, there was no statistically significant difference in performance between subjects using the dedicated and Hierarchic/Preview displays.

A number of points are suggested by these results. The dedicated display simulated a traditional control room. Fairly low level information, i.e., data, was presented in parallel, forcing the operator to build a mental model of the system which guided the selection and integration of the various pieces of displayed data required for control decisions. Integrated displays involve two models of the system: a computer model which controls the displayed information and the operator's internal model. In order to obtain needed information and to effectively control the system, the operator's model and the model underpinning the display must be compatible. Mismatched models require the operator to engage in information searches which are time consuming and mentally fatiguing, and likely to lead to a degradation in performance.

A possible conclusion which could be drawn from the results of this experiment is that the model of the system used to design the Grouped Primitive information display was a poor model, not compatible with the user's model; for example, the specific control functions may not have constituted a natural decomposition of the control activities of the system. The performance of subjects using the Hierarchic/Preview display, however, makes this conclusion suspect. Both of the integrated information

displays were based on the same basic model of the system, and this experiment revealed no performance differences between users of the dedicated display and users of the Hierarchic/Preview display. The basic difference between the two integrated displays was in the form of the displayed information. This difference suggests another explanation for the poor performance of users of the Grouped Primitive display.

Integrated information displays require the operator to call up or locate pieces of information. Thus, the operator must learn to use a set of information controls in addition to the set of system controls. The control of information, like the control of the system, requires time and mental effort. In order to be a positive contribution to a modern control room, integrated information displays must result in operator performance which is at least as efficient as that using a traditional dedicated display console. In order to compensate for the time and effort required to obtain information, an integrated display must decrease the information processing load associated with a dedicated display.

The Grouped Primitive and Hierarchic/Preview displays tried to offset the effort required to obtain information by linking the sets of displayed information to the needs of standard control decisions. Thus, the computer was to offset the information selection effort required by the integrated display by partially performing the operator's traditional function of display scanning and data selection. The fact that performance with the Grouped Primitive display was significantly and uniformly inferior to performance with the dedicated display suggests that the intended offset did not occur. The mental models developed by the users of the dedicated display allowed them to more effectively scan, select, and interpret the

primitive information than the users of the Grouped Primitive display could identify the appropriate control function and interpret the displayed information. The superior performance of users of the Hierarchic/Preview display suggests that additional computer assistance is required to compensate for the extra work required of an integrated display. The context insensitive dump of all the potentially useful primitive data characteristic of the Grouped Primitive display is not sufficient. The results suggest that in order to offset the mental load created by requiring the operator to call up information, the displayed information must not only be preselected, but it must also be integrated and presented in a form more compatible with the operator's high level information needs. Computer assisted scanning, selection, and integration seem to be required to offset the effort required to request information and the reorientation necessitated by the variable console contents.

The lack of a difference in performance between the users of the dedicated display and the users of the Hierarchic/Preview display suggests that the discrete control model which guided the design of the display was a model which was similar to, or at least compatible with, the users' internal models of the system. The strategy of keying information to control functions and of dynamically reinterpreting lower level information into percept-like chunks appears to be an adequate compensation for the additional operator effort required by the integrated display. The combination of these attributes makes the "artificial", computer assisted display competitive with the "natural", operator controlled search, select, and integrate information acquisition activity required by conventional display panels.

The low error rate for all the displays was interesting. At least for this system, it appeared to make little difference which display was

used. Subjects consistently seemed to understand the system constraints, and to seek the information needed to ensure that the contemplated control action was acceptable.

The displays did effect the subjects' ability to process engines through the system. Session score depended on few errors, and either a great many engines of some type, or a few less engines with more of the expedited types. The results of the multiple comparison tests indicate that significantly more expedited engines were processed by users of the dedicated display than by those using the Grouped Primitive display; the number of expedited engines by users of the Hierarchic/Preview display fell between the two other displays. It is possible that the dedicated display gave the operator a slight edge because the full contents of the Buffer Storage area were continuously displayed, enabling users to reflect on expedite strategies during slow periods. Several of the comments from subjects using the Hierarchic/Preview display indicated that it would have been helpful to have had the full description of the order and contents of Buffer Storage more easily accessible.

As the dedicated and Hierarchic/Preview displays showed no difference in scores, it should be no surprise that they showed no difference in either total number of engines processed or the number of expedited engines processed; both displays resulted in significantly more engines processed than users of the Grouped Primitive display.

The resource utilization measures, BUSY and HELD, are strongly related to the overall operator performance; the comparison of displays for these measures follows the same pattern as that for session score.

One note is needed about the outliers. In most cases, extreme values were associated with users of the Grouped Primitive display. This



display tended to tax a marginally skilled operator. In the debriefing sessions, subjects using this display, particularly those with lower average scores, mentioned the disorienting quality due to the variable contents of the information console. Grouped Primitive display stated that they found the display difficult to get used to. One subject who used it became so frustrated that he relied on the display all query procedures for well over fifty percent of his information. Integrated displays burden the users in that they require spatial re-encoding of the displayed information when the screen contents change. The nature of the dedicated display meant that this re-encoding was not required so that information was always in the same place. For example, experienced subjects were likely to remember where on the screen the state of Station 1 would be displayed and to simply read the state of the component in order to check its status. For the integrated display users, however, the subjects probably had to read the system name as well as its state, e.g. STATION 1: CLEAR. For several of the control functions, the Grouped Primitive Display presented the status of as many as eight or nine system components. This volume of information was high for quick and effortless re-encoding, perception, and integrating. Although the Hierarchic/Preview display also required spatial re-encoding, the limited amount of displayed information and its synthesized form seem to result in fewer assimilation and integration problems, as evidenced by the performance of users of this display type. In general, the only negative comments about the display format or contents volunteered by subjects at the debriefing session were from users of the Grouped Primitive display.

The discrete event log for each session was examined to obtain a tally of the numbers and types of information queries made by users of the

two integrated displays. The users of the Grouped Primitive display tended to circumvent the design principles on which the display was based. They usually relied on three or four query buttons for the majority of their information. Typically, the buttons associated with Buffer Control, Entry Control, and Test Exit Control were heavily used, the button for Repair Exit Control Information was occasionally used, and the other functionally related buttons were largely ignored. One reason given for this was the overlap between the various sets of information. Harking back to the disorienting quality of the display, several subjects said it was easier to get used to three or four different displays rather than seven. Evidently the time required to obtain information with the "display all" procedures or the amount of information displayed on the screen by a "display all" request discouraged Grouped Primitive users from using it frequently. Reliance on the "display all" display would have eliminated some of the objections about the difficulty in interpreting the different sets of displayed information. With one exception, however, none of the users of the Grouped Primitive display relied extensively on it.

Users of the Hierarchic/Preview display utilized the information console in patterns more consistent with those expected, given the design. These users rarely used the lower level, primitive information available with the shift procedure. The most frequently used shift procedures were with Buffer Control and Repair Entrance Control. These buttons provided information about the complete contents, order, engine type, and test status of the Buffer Storage and Temporary Storage queues, respectively. The "display all" button was used a bit more by subjects with the Hierarchic/Preview display. When questioned, the majority indicated that

they used it to obtain information about the storage area queues. One person used the display all button merely to fill the time between control actions; when he required information to evaluate the feasibility of a control action, he always queried the button corresponding to the appropriate control function.

Several users of the dedicated display commented on potentially fatiguing qualities of the static display. They felt that though it was not a burden for a thirty minute control session, they would not want to control the system for long periods of time; the boredom was wearing. A number asked if they could bring in magazines, friends, etc. to help pass the time. None of the subjects using the integrated displays indicated any boredom. It is possible that the additional control actions required by the integrated display helped to relieve the monotony of the control task.

At the debriefing which concluded the last experimental session, subjects were asked to describe their control strategies. To focus the discussion, subjects were asked to describe the procedures employed to fill Station 2, for feeding and removing engines from Station 6, and for handling failed engines. Every subject stated that it was his or her goal to keep Station 6 filled as much of the time as possible. Most of the very successful operators, held an engine at Station 6 until an engine needing test was waiting at Station 4. Those that released an engine as soon as it was tested, regardless of the state of Station 4, did not perform well.

Subjects varied widely in their policies for repairing failed engines. Some repaired all engines, others repaired only engines which were of the type being expedited, and others repaired no engines until Temporary

Storage and Station 5 were both full. None of these strategies, taken in isolation, seemed to result in consistently superior performances.

Subjects also varied widely on their policies for filling Station 2. Some, immediately after releasing an engine from Station 2, routed another engine to the station. Others waited for periods, up to some upper limit, in the hope that they could route an expedited engine from Station 1 to Station 2. One or two subjects allowed Station 2 to remain idle indefinitely, until an expedited engine could be routed to the station. A number of subjects conditioned their policy on the contents of the Repair Station; if Station 7 had an expedited engine waiting, they would be more likely to leave Station 2 clear until an expedited engine was available. Generally, the more successful operators had strategies which kept Station 2 filled, or would only wait a short period of time before routing a nonexpedited engine to the station. Strategies in which controllers consistently sacrificed throughput in favor of a higher number of expedited engines tended to result in lower scores.

With the exception of the policy for holding engines, the interviews with the subjects did not identify highly effective strategies or strategies which were display related. The strategies sounded very similar, but the variability in performance belies this conclusion. It would be interesting to examine the discrete event log for the control sessions and infer strategies; it seems likely that highly effective operators had fairly unique strategies.

This study was undertaken in order to demonstrate the use of the discrete control modelling methods in designing integrated information displays and to test the effect of the resultant displays on operator performance. The experiment showed that for at least one of the integrated

displays, operator performance was no worse with an integrated display than with a dedicated display. This is an important result for space limited control stations where growing information needs are necessitating computer based, integrated information systems.

The significant differences in performance between users of the Grouped Primitive display and the users of both the dedicated and Hierarchic/Preview display suggest that a poorly designed integrated display can seriously degrade performance. The differences between the integrated displays suggest attributes of integrated displays which enhance user performance. The Grouped Primitive display attempted to compensate for the additional effort required by user controlled information by using system alerts to notify the user of events potentially requiring operator attention and by diagnostic messages to assist in unlocking the system. These enhancements were evidently not a sufficient offset. On the other hand, the addition of preview information and the reformulated, higher level information of the Hierarchic/Preview display did seem to compensate for the additional effort. Integrated displays, in order to make up for the additional effort required to use them, must not only selectively display information, but must also preprocess the information, presenting it in a form easily assimilated by the operator.

A comparison of the differences in performance between users of the Grouped Primitive and Hierarchic/Preview displays add another important insight. It is likely that the tendency in the computer based, integrated display developed in the future will be for them to look more like the Grouped Primitive display than the Hierarchic/Preview display; that is, it is likely that a typical integrated display will provide low level data, similar to that now presented in dedicated form but in a centralized,

delimited way. This research suggests that the result would be a serious degradation in operator performance. A good deal of care must be taken in designing integrated displays. Computer based displays afford a great deal of flexibility in the display of information; this flexibility must be creatively exploited so that information is pre-processed and presented in forms compatible with the user's needs, rather than in forms directly corresponding to the individual hardware components.

Another insight concerned the potentially positive contribution that integrated, operator controlled information displays can make to the boredom inherent in some control tasks. Particularly in more supervisory tasks, modern controllers may be required to be alert and watchful during long periods of inactivity. The operator actions required by an integrated information display may add a valuable dimension to the conventional control room where slowly changing displays can potentially mesmerize the operator.

This research has made two important contributions to the area of display design. Perhaps the most important concerns the relative contributions to observed performance variance of individual subjects and displays. The percentage of performance variance attributable to subject variation far outweighed the portion attributable to the differences in displays. It seems that the most important factor in improving the human-machine interface is the selection of a "good" operator. A well-designed display enhances the performance of a highly skilled operator; a poorly designed display further degrades the performance of a less skilled operator.

The second major contribution made by this research is the demonstration of the utility of the discrete control modelling methods in designing displays. To date, design of integrated displays has been system specific

and fairly intuitive, suffering from a lack of well-defined, repeatable, and tested procedures for creating computer based information systems. No coherent set of standards or principles exists to guide the design of integrated displays. Yet, it is widely agreed, and this experiment confirmed, that a poorly designed integrated display results in significantly worse performance than a dedicated display. This research demonstrated the efficacy of one approach to designing displays which organized information by the needs of the users rather than on hardware-related logic. The hierarchic structure of the modelling approach suggested was to exploit the capabilities of computer based information systems by allowing variable format, and dynamic levels of aggregation of lower level data. This modelling method constitutes one, coherent, well-defined theoretical approach to the design of integrated information systems. Creative and rigorous design principles for information displays will improve the human-machine system and help to reinstate the human operator as a vital and strong link. Much more research is required, however, to fully exploit the potential which computer based displays afford.

## REFERENCES

Jagacinski, R. J., R. A. Miller, W. W. Johnson, and R. B. Nalavade, "Coordinative and Strategic Aspects of Discontinuous Tracking", The Ohio State University Research Foundation, 1981.

Miller, R. A., "Identification of Finite State Models of Human Operators", The Ohio State University, Systems Research Group, 1979.

Miller, R. A. and C. M. Mitchell, "A System Oriented Approach to Information Display", Proceedings of the 1981 International Conference on Cybernetics and Society, Boston, MA, pp. 723-727.

Mitchell, C. M., "The Design of Computer Based Integrated Information Displays", Ph.D. Dissertation, The Ohio State University, 1980.

Roscoe, S. R., and J. E. Eisele, "Integrated Computer-Generated Cockpit Displays", in Monitoring Behavior and Supervisory Control, T. B. Sheridan and G. Johannsen (eds.), Plenum Press, New York, 1976.

Sevenler, K., "The Use of Structural Information to Compute Subsystem States in Discrete Control Modelling", M.S. Thesis, The Ohio State University, 1980.

Sheridan, T. B., and G. Johannsen (eds.), Monitoring Behavior and Supervisory Control, Plenum Press, New York, 1976.

Singleton, W. T., "The Model Supervisor Dilemma", in Monitoring Behavior and Supervisory Control, T. B. Sheridan and G. Johannsen (eds.), Plenum Press, New York, 1976.